VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Testování DOM parserů

DOM Parsers Testing

2013 Michael Zhasil

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michael Zhasil**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Testování DOM parserů
DOM Parsers Testing

Zásady pro vypracování:

V dnešní době existují dva typy parserů: SAX a DOM. Parsery typu DOM používají stromově založený přístup k navigacii v XML dokumentu. Parser zpracuje XML data a vytvoří objektově orientovanou hierarchickou reprezentaci dokumentu. Úkolem této práce je testování různych implementací specifikace DOM.

Úkoly:
1. Nastudujte specifikaci DOM.
2. Vyberte co největší počet implementací DOM.
3. Proveďte testy pro různé kolekce.
4. Výsledky experimentů vyhodnoťte.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Chovanec**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA 1

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2. května 2013 ..Zkasil.....

Rád bych touto cestou vyjádřil poděkování mému vedoucímu bakalářské práce Ing. Peteru Chovancovi, za jeho veškerou pomoci, usílí a čas, který mi věnoval při tvorbě této práce.

Abstrakt

Cílem této práce porovnání výkonnosti a paměťové náročnosti objektově orientových XML parserů.
V první části je popis jazyka XML, jeho historie a využití v součastnosti a také náhled na jazyky od XML odvozené.
Dále budou předtsaveny nejen objektově orientované technologie zabývající se čtením, úpravou a zápisem XML dokumentů. Porovnány budou výhody a nevýhody jednotlivých přístupů při nasazení v různých prostředích
Klíčová slova: DOM, XML, .NET, Java, testy, parsování

Abstrakt

Goal of this thesis is to compare performance and memory requirments of object oriented XML parsers.
First part of this thesis contains description of XML, it´s history, how it´s useful nowadays and also brief description of languages derived from the XML language.
Later on will introdouced also not object oriented technologies for reading, editing and creating of XML documents. The will be comparation of pros and cons of each individual access at different environments
Keywords: DOM, XML, .NET, Java, testing, parsing

Seznam použitých zkratek a symbolů

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
BSD	Berkeley Software Distribution
CSS	Cascading Style Sheets
DOM	Document Object Model
DTD	Doctype
EBI	European Bioinformatics Institute
EPA	Enviromental Procetion Agency
FDCCI	Federal Data Center Consolidation Initiative
GML	Generalized Markup Language
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HW	HardWare
IBM	International Business Machines Corporation
ISO	International Organization for Standardization
J2ME	Java Platform, Micro Edition
JAXP	Java API for XML Processing
JCP	Java Community Process
JIPID	Japan International Protein Information Database
KMZ	Keyhole Markup Language
MathML	Mathematical Markup Language
MIPS	Munich Information Center for Protein Sequences
MSXML	Microsoft XML Core Services
OOP	Object-oriented programming,
PDF	Portable Document Format
PHP	Hypertext Preprocessor
PIR-PSD	Protein Information Resource - Protein Sequence Database
RDF	Resource Description Framework
RSS	RDF Site Summary
RTF	Rich Text Format
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SIB	Swiss Institute of Bioinformatics
SMIL	Synchronized Multimedia Integration Language

SOAP	Simple Object Access Protocol
Stax	Streaming API for XML
SVG	Scalable Vector Graphics
SW	SoftWare
USA	United States of America
UTF	Unicode Transformation Formats
VTD	Virtual Token Descriptor
W3C	World Wide Web Consortium
WML	Wireless Markup Language
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XSLT	Extensible Stylesheet Language Transformations

Obsah

Seznam tabulek

Seznam obrázků

Kapitola 1

Historie jazyka XML

Jazyk XML[55] je podmnožinou jazyka SGML[56] (Standard Generalized Markup Language, ISO 8879:1986). Jazyk SGML vznikl v roce 1986 jako nástupce jazyka GML[57], který byl vytvořen v 60. letech firmou IBM. Z jazyka SGML vychází, krom XML, také jazyky HTML[28] a XHTML[31]. Jazyk SGML byl využíván vládními insitucemi pro ukládání velkých strojově čitelných dokumentů, tyto dokumenty musí být často čitelné i po mnoha letech. Jazyk SGML byl postupně rozšiřován o další funkce a možnosti. Dnes rozlišujeme 3 hlavní verze.

Původní SGML z roku 1986.

SGML(ENR) - Extended Namig Rules z roku 1996 umoňující použití libovolných značek.

SGML(ENR+WWW) - verze z roku 1998 zaměřena na lepší podporu požadavků kladených WWW a XML

Jazyk SGML byl používán zejména v odborných a vládních kruzích mezi širší veřejnost se dostal až jazyk XML, který vznikl v roce 1998.

Jazyk XML vznikal jako technologie, která má být jednoduší variantou k poměrně komplexnímu SGML avšak zachová kvality původního jazyka.

Pro vznik XML jazyka byly definovány následující požadavky:

- Validovatelnost
- Strukturovatelnost
- Nezávislost na platmofmě
- Rozšiřitelnost
- Nezávislost na médiu
- Kompatibilata s SGML - SGML parser dokáže pracovat s XML dokumenty v opačném směru nastávají problémy kvůli zjednodušení XML
- Stručnost
- Čitelnost
- Minimalizovat počet volitelných funkcí

Samotný jazyk XML byl standardizován v roce 1996 a to ve verzi 1.0, verze 1.0 je byla dále rozvíjena i po uvedení XML 1.1 v roce 2006, naspoledy bylo XML 1.0 revidováno v roce 2008.

Kapitola 2

Jazyk XML

2.1 Základy

XML je podobně jako HTML značkovací jazyk, hlavní rozdíl mezi oběma jazyky tkví v tom, že zatímco tagy HTML udávají následný vzhled stránky respektive obsahu tagu byl jazyk XML vytvořen k popisu dat a k jejich přehlednému, logickému a hierarchickému uspořádání, což také umožňuje následnou jednoduchou strojovou editaci nebo čtení.

```
    <?xml version=”1.0”encoding=”UTF-8”>
<clanky>
<!— Root Element —>
<clanek id=”1”>
    <hlavicka>
        <autor> Tonda Vrba< /autor>
        <datum>10.10.2010< /datum>
        <mail>vrba@mail.cz</mail>
    </hlavicka>
    <obsah>
        <nadpis>Jazyk XML</nadpis>
        <text>Poslední revize XML 1.0 proběhla v roce 2008.</text>
    </obsah>
</clanek>
<clanky>
```

Příklad ukazuje jednoduchý XML ”Well formed”dokument s hierarchickou strukturou elementů. Na příkladu popíši co musí XML dokument splňovat.
XML dokument musí obsahovat hlavičku, kde je zapsána verze a znakovaná sada.
Musí existovat jeden Root element, který obsahuje všechny ostatní elementy.
Standard je znaková sada ISO 10546 - zcela shodná s Unicode
Kódování UTF-16 a UTF-8 - standard vyžaduje aby tyto sady podporovaly všechny aplikace
Lze použít i jiná znaková sada - musí být uvedena v deklaraci
Každý tag je párový, jednolivé tagy se nesmí křížit.
Tagy jsou case sensitive zápis <Autor> Jmeno <\ autor> je neplatný

Název elementu může obshovat číslici nebo podrtržítko("_"), ale nesmí jimi začínat
Název nesmí začínat písmeny *XML* Element může obsahovat atributy, pokud atribut obsahuje jeho hodnota musí být uzavřena v uvozovkách. Pokud hodnota atributu obsahuje řetězec v jednoduchých uvozovkách musí být atribut uzavřený ve dvojitých uvozovkách a naopak.
Komentáře píšeme ve tvaru <!- - Text komentáře - ->

2.2 Rozdíl mezi verzí 1.0 a 1.1

XML 1.0 je zpětně kompatiilní s Unicode, XML 1.1 je dopředně kompatibilní - uživatel si může rozšířit znakovou sadu pro pojmenování elementů
XML 1.0 je konzervativní, tedy vše co není povoleno je zakázáno, XML 1.1 je oproti tomu liberální - vše je defaultně povoleno
V XML 1.1 můžou názvy elementů obsahovat dříve zakázané znaky - matematické operátory, symboly např. ©
NEL(new line) znak může být ve verzi 1.1 umístěn kdekoliv v textu

2.3 Doctype

Doctype[21] určuje pravidla psaní daného XML dokumentu, je také požadavkem pro vytvoření validního XML dokumentu. V DTD je definováno jak mají jednotlivé elementy XML dokumentu vypadat, jestli mají subelementy, jaký tvar a jakých hodnot můžou nabývat případné atributy jednotlivých elementů. Pomocí DTD v podstatě definujeme nový jazyk založený na XML, protože definujeme množinu a tvar elementů v dokumentu použitelných, čímž jim také poskytujeme určitý pevně daný vyznám. Použitím DTD také velice zjednodušujeme jak tvorbu tak samotné parsování a kontrolu správného zápisu XML dokumentu.

Pro definování počtu výskytů elementů či subelementů v dokumentu se v DTD používají následující znaky:

- Samostatné jméno elementu značí právě jeden výskyt
- Znak ? značí žádný nebo jeden výskyt
- Znak + značí jeden nebo více výskytů
- Znak * značí libolný počet výskytů

Doctype se do XML vkládá pod hlavičku ve tvaru:
<!DOCTYPE root_element SYSTEM "NazevSouboruSDTD.dtd">
Jak se doctype píše uvedu opět na příkladu:

<?xml version="1.0"encoding="UTF-8"?>
<!DOCTYPE clanky [
<!ELEMENT clanky (clanek+) >

```
<!ELEMENT clanek (hlavicka, obsah) >
<!ATTLIST clanek
              id        CDATA    #REQUIRED>
<!ELEMENT hlavicka      (autor+, datum, mail)>
<!ELEMENT autor         (#PCDATA)>
<!ELEMENT datum         (#PCDATA)>
<!ELEMENT mail          (#PCDATA)>
<!ELEMENT obsah         (nadpis, text)>
<!ELEMENT nadpis        (#PCDATA)>
<!ELEMENT text          (#PCDATA)>
] >
```

Jak lze vypozorovat z příkladu doctype opět začíná hlavičkou stejně jao samotný XML dokument, následuje otevření !DOCTYPE s názvem root elementu. U každého elementu (!ELEMENT) je uveden obsah daného elementu, ten může obsahovat subelementy - pak jsou zapsány spolu se symboly zaznamenávající četnost výskytu subelementu, nebo element obsahuje už samotná data v textové podobě (#PCDATA). Obsah elementu také upravují klíčová slova EMPTY - pro prázdný element a ANY - pro libovolný obsah elementu.
Pokud by element obsahoval jak data tak subelementy vypadal by jeho DTD zápis následovně
<!ELEMENT hlavicka (#PCDATA | autor+ | datum | mail)>
Tedy #PCDATA musí být uvedeno jako první a pro spojení skupiny se místo čárky(",") musí použít znak "|"

Definice atributu je podobně ve tvaru:
<!ATTLIST "jmeno_tagu" "atributy">
Pro libovolný řetězec v atributu slouží slovo CDATA, sekce CDATA se může vyskytnou i v samotném XML dokumentu, kde značí volný neirtepretovaný text, tedy text kde lze použít speciální znaky a tagy krom sekvence]] > která značí konec sekce CDATA
Dále atributy mohou obsahovat NMTOKEN a NMTOKENS, NMTOKEN znamená jedno slovo složené z číslic a písmen, NMTOKENS je více takovýchto slov oddělených mezerou
Atributu taky můžeme také určit výčet hodnot které jsou pro daný atribut platné. Uvedu opět na příkladu:

```
<!ATTLIST talíř
      TVAR (kulatý|hranatý)
      STAV (čistý|špinavý) "čistý">
```

Možné hodnoty jsou opět odděleny pomocí "|"pokud je za výčtem v uvozovkách uvedena jedna z možných hodnot, jedná se o defaultní hodnotu.

Další typ atrbitu slouží pro vytváření odkazů v dokumentu podle jedinečné ID hodnoty. Těmito atributy jsou

- ID

  atributy ID musí být unikátní v rámci celého dokumentu, tedy i u elementů s různými názvy, každý element může mít pouze jedno ID

- IDREF

  Podobně jako NMTOKEN IDREF obsahuje jedno ID použité v dokumentu.

- IDREFS

  Opět podobné s NMTOKENS obsahuje několik ID použitých v dokumentu oddělených pomocí mezery

Poslední částí u definice atributu je povinnost atributu

- #REQUIRED

  Atribut je povinný

- #IMPLIED

  Takto označený atribut může být vynechán

- #Fixed

  Označuje povinnou hodnotu atributu, tedy atribut nesmí nabývat žádné jiné než zadané hodnoty.

  Opět příklad zápisu: <!ATTLIST talíř CDATA #FIXED "kulatý">

```
<?xml version="1.0"encoding="UTF-8">
<!DOCTYPE clanky SYSTEM "doctype_clanky.dtd">
<clanky>
<!— Root Element —>
<clanek id="1">
    <hlavicka>
        <autor> Tonda Vrba< /autor>
        <datum>10.10.2010< /datum>
        <mail>vrba@mail.cz</mail>
    </hlavicka>
    <obsah>
        <nadpis>Jazyk XML</nadpis>
        <text>Poslední revize XML 1.0 proběhla v roce 2008.</text>
    </obsah>
</clanek>
</clanky>
```

Do XML dokumentu je následně přidán vytvořený doctype a z Well formed dokumentu se stává dokument validní.

2.4 Jazyky a technologie odvozené od XML

Jednoduchost a zároveň široké možnosti, které jazyk XML přinesl umožnil jeho velice rychlé rozšíření a použití jako základ dalších jazyků a technologií mnoho z těchto technologií je každodenně používáno enormním množstvím bežných laických uživatelů.

- XHTML

  Extensible HyperText Markup Language

  Značkovací jazyk kombinující syntaxi XML a sématiku HTML, v předkladu znamená rozšiřitelný značkovací jazyk.

  Verze odpovídají HTML, tedy Strict, Transitional a Frameset

- RSS

  RDF[6] Site Summary

  Rodina XML formátů určených pro čtení novinek z webu nabízejících RSS[5] kanál

  Vyskytuje se na webech, které jsou často aktualizovány, např. zpravodajské weby

- MathML

  Mathematical Markup Language[8]

  Značkovací jazyk vytvořen exkluzivně pro popis a zobrazení matematických symbolů a vzorců

- SVG

  Scalable Vector Graphics[9]

  Formát popisující dvourozměrnou vektorovou grafiku pomocí XML

- XMPP

  Extensible Messaging and Presence Protocol[10]

  Původně protokol pro Instant Messaging na síti Jabber

  Decentralizovaný

  Otevřené zdroje a specifikace

- Google Earth a Google Maps

  umožňuje uživatelům si své projekty ukládat ve jak ve formátu XML(KML) tak i jako komprimovanou XML (KMZ)

- SOAP

  Simple Object Access Protocol[11]

  XML zpráva přenášena na protokolu HTTP

  tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace

  volná vazba mezi serverem a klientem

- SMIL

  Synchronized Multimedia Integration Language[12]

  Pomocí tohoto jazyka můžeme vytvářet a publikovat online i offline multimediální prezentace

- WML

  Wireless Markup Language[13]

  Jazyk k vytváření jednoduchých webových stránek používaných pro zobrazení webu na tabletech a mobilních telefonech

- DocBook

  Docbook[14] je značkovací jazyk pro tvorbu dokumentace, původní zaměření byla dokumentace HW a SW

  Lze konvertovat do dalších typů souborů (PDF, HTML, RTF)

  Mimo XML je implementován také v SGML

- RDF

  Resource Description Framework

  Obecná metoda pro modelování informací v různých syntaxích

  Umožňuje explicitní a formalizovaný popis určité problematiky. např. Popsat obsah HTML stránky

  RDF způsob popisu zdrojů je jednou z hlavních komponent sémantického webu

Kapitola 3

Parsování XML

XML parsery[7] slouží jak pro čtení tak pro zápis do dokumentu. Pro rozparsování dokumentu nám stačí mít XML dokument ve Well Formed tvaru. XML parserem se také ověřuje validita dokumentu. Dnes jsou již knihovny pro XML parsing standardní součástí balíku knihoven pro Javu, C# i PHP. XML parsery dělíme na dva hlavní proudy a sice "Událostně řízené"sekvenční zpracování XML souboru. Parsery tohoto typu se zahrnují pod rodinu Simple Api for XML (SAX)[36]. Druhým typem přístupu je vytvoření stromové struktury objektů v paměti reprezentujících XML soubor, takto vytvořené parsery spadají pod Document Object Model(DOM)[1].

Při parsování Validního dokumentu je jako první provedena validace, tedy analýza dokumentu, kontrola syntaxe, ověření, že si dokument odpovídá s DTD. Dále se načtou defaultní hodnoty atributů nejsou li tyto hodnoty zadány jinak. Na závěr se dokument převede na interní datovou specifikaci se kterou je možno dále pracovat.

3.1 DOM

Byl vytvořen konsorciem W3C[2] pro sjednocení přístupu a manipulace s HTML elementy, bez jazykové či platformové závislosti. K dokumentům zpracovávaným pomocí DOM přistupujeme jako ke stromu, každý element dokumentu odpovídá jednomu uzlu stromu. Vzhledem k tomu, že celý dokument je nahrán do paměti je následné procházení a modifikování dokumentu velice jednoduché. Použití DOM parseru je výhodné pokud budeme k prvkům přistupovat opakovaně, nebo bude výběr prvků náhodný. DOM obsahuje 3 úrovně specifikace, platí že každý dokument psaný pro úroveň menší než nejnižší musí splňovat všechny požadavky úrovně pro kterou je tvořen a také všechny požadavky všech předchozích úrovní.

3.1.1 Úrovně DOM

- Level 1

  Core - poskytuje jak nízkoúrovňovou sadu základních rozhraní která mohou reprezentovat jakýkoliv strukturovaný dokument tak i sadu rozšiřujících rozhraní pro sestavení libovolného XML dokumentu

HTML - poskytuje vysokoúrovňová rozhraní pro prácí s HTML dokumenty. Implementace HTML DOM Level 1 zahrnuje i základní rozhraní z Core

- Level 2

  Core

  - Rozšiřuje rozhraní z Level 1
  - Přidává rohzhraní určené výhradně pro práci s XML
  - Představena metoda getElementById
  - Přidány jmené prostory a nástroje pro jejich obsluhu

  Views

  - Umožňuje dynamický přístup a úpravu obsahu
  - Rozhraní AbstractView a DocumentView

  Events

  - Přidává události a jejich obsluhu.
  - Ještě neobsahuje obsluhu událostí po stisku klávesy

  Style / CSS

  - Přidává rozhraní pro práci s grafickými styly
  - Možnost dynamické úpravy stylů

  Traversal and Range

  - Možnost dynamického procházení dokumentu
  - Možnost hromadné úpravy uzlů - podle zadaného rozsahu(Výběr všeho mezi bodem A a bodem B)

  HTML

  - Rozšiřuje možnosti z předchozí verze
  - Přidává možnost dynamického přístupu do dokumentu a jeho úpravy

- Level 3

  Core

  - Rozšiřuje možnosti z předchozí verze

  Load and Save

  - Umožňuje dynamicky načíst obsah XML dokumentu do dokumentu typu DOM a poté provést serializaci DOM dokumentu do XML

  Validation

– Dynamická úprava obsahu a struktury dokumentu aby se zajistila jeho validita vůči použitému DTD

Xpath

– Umožňuje pro přístup do uzlů používat Xpath[15]

3.2 SAX

Dokument je rozdělen na jednotlivé části a poté čten sekvenčně, při nalezení a přečtení jednotlivých částí je vyvolána událost, která je obsloužena programátorem. Dokument nemusí být při čtení celý uložen v paměti i validace dokumentu probíhá postupně po jednotlivých částech. SAX není standardizován v konsorciu W3C ani v jiné standardizační organizaci, ačkoliv je tento přístup k dokumentům vysoce rozšířen a podporován. za vývojen Simple Api for XML a rozšířením stojí skupina XML-DEV

3.3 Rozdíly mezi DOM a SAX

- DOM

  Před zahájením jakýkoliv operací s dokumentem se musí dokument nejprve celý projít a zpracovat

  Celý dokument je nahrán do paměti

  Můžeme provádět modifikace dat i struktury

  Dokument lze procházet oběma směry.

- SAX

  Dokument je zpřístupňován postupně

  V paměti je vždy nahrána jen část dokumentu

  Nelze provádět modifikace

  Procházíme pouze odshora dolů, chceme li se vrátit musíme opakovat celý průchod

Kapitola 4

Seznamy parserů

4.1 Seznam použitých parserů

4.1.1 DOM4J 1.6.1

Open-Source Java parser vydaný pod BSD licencí. DOM4J[37] může zpracovat XPath, XML a XLST data. Parser plně podporuje DOM i SAX. práci s dokumentem. V testu použitá verze 1.6.1 byla vydána v květnu 2005. V současnosti je na domovské stránce projektuk dispozici také verze 2.0 ve stádiu Alpha 2.

4.1.2 JDOM 2.0.4

Open-Source parser vytvořený výslovně pro Javu a JDOM[34] je zařazen do Java Community Process (JCP), čímž se stává součástí specifikace Javy. JDOM využívá jak DOM tak SAX přístup a může tyto metody kombinovat. Například použít SAX čtení a výsledky odesílat do DOM stromu.

4.1.3 MicroDOM 2.1

Parser vydaný pod BSD licencí je podmnožinou W3C DOM API vytvořenou tak aby lépe odpovíal potřebám specifikace SVG 1.1. Cílem vývojářů MircoDOM[38] je minimalizovat počet rozhraní nutných pro přístup k datům. Pro snížení nároků na operační paměť je v parseru použity pouze nejnutnější části DOM jádra

4.1.4 Oracle DOM parser 1.4

Je součástí JAXP[39] rozhraní, poslední verze JAXP 1.4.6 byla vydána v dubnu roku 2012. Rozhraní JAXP je implementováno přímo ve specifikacích Javy, využívá podobně jako JDOM možnost přechodu či kombinování metod přístupu k XML dokument. JAXP podpuruje verzi XML 1.0 i XML 1.1. DOM parser zcela vyhovuje specifikacím DOM Level 3

4.1.5 Saxon HE 9.4

Saxon[40] parser se od verze 9.2 vydává ve 4 verzích

- Saxon-HE (Home Edition) - open source verze vydávána pod Mozilla public licencí. Tato verze je dostupná jak pro javu tak pro .NET a poskytuje podporu XSLT 2.0, XQuery 1.0, a XPath 2.0
- Saxon-PE (Professional Edition) - rozšířená komerční verze Saxon HE, tato edice přidává možnost lokalizace do jiných jazyků, XQuery 3.0, jednoduší možnosti nastavení a také podporuje externí modely objektů jako jsou JDOM, XOM a DOM4J
- Saxon-EE (Enterprise Edition) - Dále rozšiřuje verzi professional edition o další funkcionality
- Saxon-CE (Client Edition) - určeno pro prohlížeče, kód parseru je převeden do Javascriptu pomocí Google Web Toolkitu.

4.1.6 Uxparser 3.0.79

Také označován jako Java Micro XML Parser[41] vydán pod Public licencí. Pro parser je dostačující J2ME platforma. Pro parsování dokumentů využívá parser kombinaci DOM a Pull přístupu. Tento parser je vytvořen s důrazem na jednoduchost použití a snadné zakomponování do aplikace

4.1.7 X2S 0.1.1

X2S[43] je Open Source parser bez plné podpory Xpath a XQuery, zaměřený na získání všech atributů a hodnot z XML dokumentů. Poslední update tohoto parseru proběhl v prosinci 2012. Jak je patrné z čísla verze parser je stále ve vývoji

4.1.8 Xerces 2.9.0

Tento parser je souběžně vyvíjený pro C++, Perl a Javu. Xerces[44] je sbírka několika knihoven pro manipulaci s XML soubory. V Xercesu je implementováno několik rozhraní pro práci s XML - DOM, SAX, StAX a JAXP. Xerces parser od verze 2.0 představil rozhraní Xerces Native Interface(XNI). Toto rozhraní sjednocuje přístup k XML dokumentu ať už uživatel používá jakékoliv z výše zmíněných rozhraní pro práci s XML.

4.1.9 Ximpleware 2.11

Parser používá binární formát VTD(Virtual Token Descriptor), token je kódován jako číslo ve formátu 32bitového integeru.
Ximpleware[42] je vyvíjen zároveň pro Javu, C, C++ a C#. Tento parser dokáže zpracovat pomocí XPath soubory o velikosti až 256GB.

4.1.10 XOM 1.2.9

XOM[29] parser je založený na korektnosti čtených a tvořených XML dokumentů. Nelze v něm přečíst/zapsat dokument, který neodpovídá Well-formed dokumentu. Tento parser také umožňuje současné čtení a zápisu jednoho dokumentu. Při parosvání dokumentů XOM používá SAX metodu čtení, kde čtená data zapisuje do stromové struktury.

4.2 Seznam nepoužitých parserů

4.2.1 BareXML

Tento v C napsaný parser generuje jednoduchou stromovou strukturu pro Perl skze XS rozhraní,BareXML[45] je také součástí linuxových distribucí openSuse, Debian a také součástí operačního systému pro PDA Sharp Zaurus

Přidání dalšího rozhraní pro komunikaci s parserem by ovlivnilo výsledky testovaní, proto nebyl tento parser zařazen do testování.

4.2.2 ChilkatXML

ChilkatXML[46] je možno použít v komerčních i nekomerčních aplikací a pro weby. Parser je založený na zjednodušeném DOM API. Chilkat dokáže zpracovat jakékoliv jazykové kódování. Parser má integrované možnosti Zip komprese a AES šifrování. ChilkatXML je možno použít prostředích .NET, ActiveX, ASP, C++

Parsery postavené na zjednodušené verzi jsou v testovací aplikaci zahrnuty a proto nebyl tento parser použit.

4.2.3 CougarXML

Parser napsaný v JavaScriptu využívající OOP. XML soubor je při parsování uložen jako objekt nad kterým můžouu být prováděny operace pomocí API splňující DOM level 3 standard

Projekt CougarXML[47] je s nejvyšší pravděpodobností zastavený a proto nebyl tento parser použit.

4.2.4 Crimson

Parser[48] založený na Sun X Parseru v současnosti součástí projektu Apache Attic a nahrazen Xerces parserem. Vývoj Crimsonu byl ukončen v roce 2010. Zdrojové kódy jsou stále ke stažení na webu apache.org. Crimson ve své době podporoval DOM level 2, JAXP a SAX

Tento parser nebyl použit z důvodu ukončení vývoje a nahrazení Xerces parserem

4.2.5 Microsoft XML Core Services (MSXML)

Jedná se o skupinu služeb umožňující aplikacím v JScriptu, VBScriptu a vývojových nástrojích od Microsoftu s použitím Component Object Model vytvářet aplikace založené

na XML. MSXML[49] podporuje XML 1.0, DOM, SAX, XSLT 1.0 a XML Schema. Microsoft nepodporuje MSXML v .NET aplikacích, kde má být používány pouze nástroje z knihovny System.xml. Kvůli omezení nasazení MSXML nebyl tento parser testován.

4.2.6 NanoXML

Jedná se malý rozvětvený Java parser, vytvořený též ve verzi pro mobilní telefony. Jako jednotlivé větve se považují kolekce tříd vybrané ze stromu zdrojových kódů. Jednotlivé větve se NanoXML parseru[54] se dělí na:

- NanoXML/Java - původní parser založený na DOM
- NanoXML/SAX - jako všechny další větve založený na NanoXML/Java, tato větev je předělána na podporu SAX
- NanoXML/Lite - minimalistická verze, osekaná o některé funkce například podpora DTD na druhou stranu se NanoXML/Lite může pyšnit velikostí pouchých 6KB

NanoXML bohužel nepodporuje výběr složky s XML souborem. Kvůli tomuto omezení nebyl zahrnut do testovací aplikace, která umožňuje uživatelský výběr XML souboru.

4.2.7 TinyXML

Minimalistický parser pro jazyk C++ parsuje XML soubory do stromové struktury. Tento parser nezpracovává DTD a soubory závislé na DTD nejsou doporučeny k parsování v TinyXML[50]. TinyXML také nepodporuje jmené prostory a XSLT. Ze znakových sad TinyXML přijímá pouze UTF-8 a nespecifikovanou formu ASCII podobnou formátu Latin-1.

Kvůli omezením parseru nebyl tento parser použit v testech.

Kapitola 5

Seznamy kolekcí

5.1 Divadelní hry Williama Shakespeara

Několik náhodně vybraných scénářů Shakespearových divadelních her[16] z kolekce, kterou vytvořil Jon Bosak(vedoucí vývoje jazyka XML), všechny hry jsou validovány oproti dokumentu *play.dtd*. Hry byly testovány jako samostatné soubory tak i jako kolekce souborů.
Kolekce obsahuje tyto hry:

- All's Well That Ends Well - Velikost souboru: 212kB
- As You Like It - Velikost souboru: 194kB
- Romeo and Juliet - Velikost souboru: 221kB
- Twelfth Night - Velikost souboru: 189kB

5.2 EPA Geospatial Data Access Project

S cílem zlepšení stavu životního prostředí a zdravotního stavu populace shromažďuje EPA(Enviromental Protection Agency)[17] informace o lokalitách a zařízeních na která se vztahují ekologické regulační předpisy. Soubor také obsahuje klíčové informace o evidovaných objektech, jako je poloha nebo oblast ekologických zájmů.
Velikost souboru: 227 802kB

5.3 Federal Data Center Consolidation Initiative (FDCCI)

Databáze datových center v USA. Cílem FDCCI[18] je uzavření 40% datových center do konce roku 2015. Práce prováděná v těchto 40% se má rozdělit mezi zbývající datová centra s cílem sjednocení práce. Soubor obsahuje informace o Názvu oddělení kterému datové centrum podléhá, název centra, poloha datového centra(město, stát, GPS souřadnice) a stav datového centra
Velikost souboru: 472kB

5.4 Inex-Wiki(Offline Wikipedia)

Offline soubory Wikipedie[51] jsou rozděleny na části do podsložek, každá podsložka obsahuje 30 000 souborů. Soubory mají různu velikost v rozmezí od 500kB - 1kB. Všechny soubory jsou jako u online wikipedie svázány mezi sebou pomocí odkazů. Poměrně se v textu dřídají i různé znakové sady jako latinka, azbuka nebo čínské písmo.
Velikost kolekce 10 000 souborů: 161MB
Velikost kolekce 60 000 souborů: 930MB

5.5 PIR-PSD

Jedná se mezinárodní databázi proteinů, její součástí je i databáze SwissProt [53]. Celá databáze je založena na Atlas of Protein Sequence and Structure (1965-1978). V současnosti jde o spojení databází z těchto zdrojů:

- MIPS (Munich Information Center for Protein Sequences)
- JIPID (Japan International Protein Information Database)
- EBI (European Bioinformatics Institute)
- SIB (Swiss Institute of Bioinformatics)

PIR-PSD[20] je součástí konsorcia UniProt a její data jsou integrována do UniProt Knowledgebase. Testovaná verze obsahuje 262525 sekvencí proteinů a 89717977 reziduí Původní PIR byl vytvořen americkou nadací National Biomedical Research Foundation.
Velikost souboru: 700MB

5.6 Soubor čísel

Soubor obsahující 2147483 náhodně vygenerovaných čísel. Tento soubor je pouze Well-Formed a neobsahuje žádné jiné informace.
Velikost souboru: 43,6MB

5.7 SwissProt

Databáze proteinů vytvořená v Swiss Institute of Bioinformatics, je součástí UniProt konsorcia a PIR-PSD databáze. Databáze obsahuje popis funkce proteinu, strukturu proteinu, jeho možné variace a modifikace. Při tvorbě databáze se autoři snaží o co nejvyšší možnou spolupráci a začlenění údajů z jiných databázi, stejně tak jako minimalizovat počty opakujících se informací.
Velikost souboru: 112,1MB

Kapitola 6

Výsledky testů

Všechny testy byly prováděny na tomto zažízení:

Typ:	Server
Operační systém:	Windows Server 20008 R2 Datacenter, Service Pack 1, 64-bit
Procesor:	Intel Xeon E5-2690, 2,90 GHz
Operační paměť:	288GB

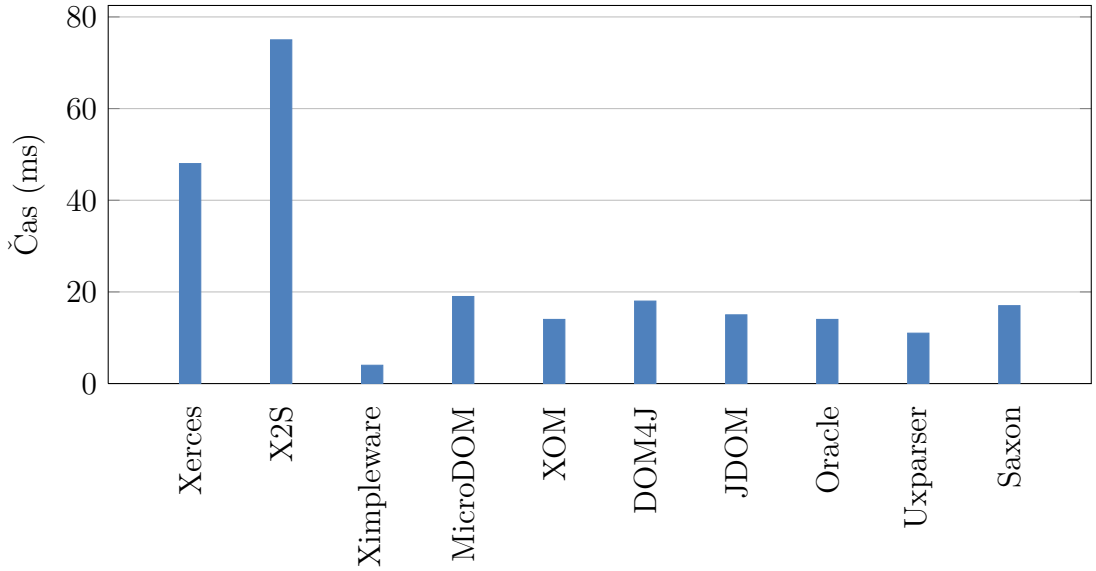
6.1 Divadelní hry Williama Shakespeara

6.1.1 Hry jako samostatné soubory

All's Well That Ends Well

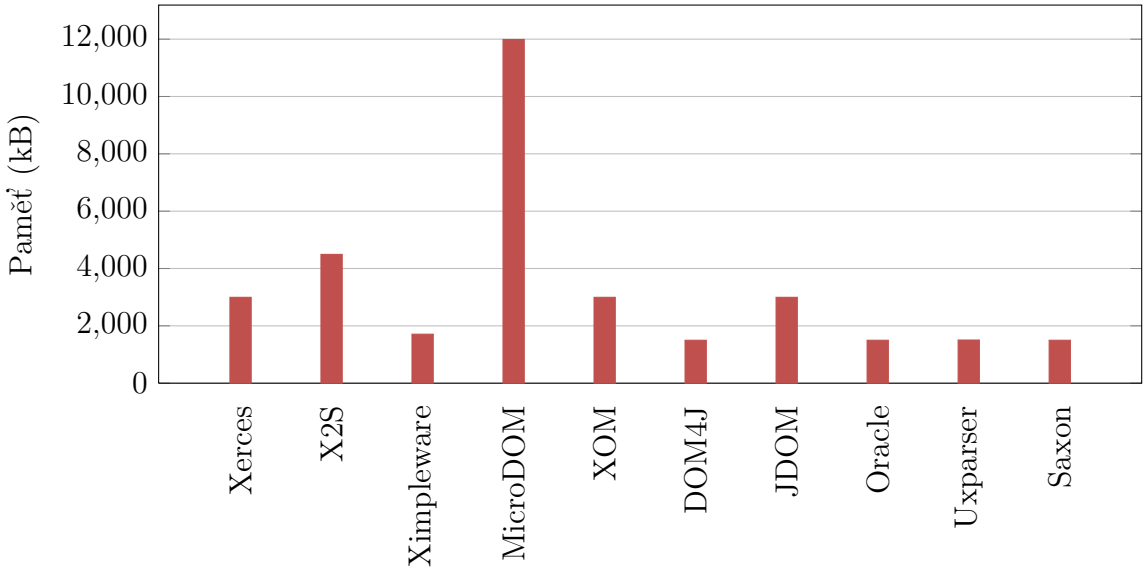
Jméno parseru	**Doba zpracování(ms)**	**Využitá paměť(kB)**	**BitRate (kB/s)**
Xerces	48	2 996	4 412
X2S	75	4 493	2 823
Ximpleware	4	1 710	52 939
MicroDOM	19	11 990	11 145
XOM	14	2 996	15 125
DOM4J	18	1 498	11 764
JDOM	15	2 996	14 117
Oracle DOM	14	1 498	15 125
Uxparser	11	1 506	19 251
Saxon	17	1 498	12 456

Tabulka 6.1: Hodnoty hry: All's Well That Ends Well



Obrázek 6.1: Doba zpracování All's Well That Ends Well

Jak lze vidět z grafu a tabulky 6.1 parser Ximpleware se jako jediný dostal po hranici 10ms, následován druhým nejrychlejším Uxparserem s časem 11ms. Většina ostatních parserů skončila s testováním v rozmezí 14ms - 19ms. Jako nejhorší v tomto testu skončila dvojice Xerces a X2S, tyto parsery skončily se svými časy 48ms a 75ms daleko za průměrem.



Obrázek 6.2: All's Well That Ends Well: Využitá paměť

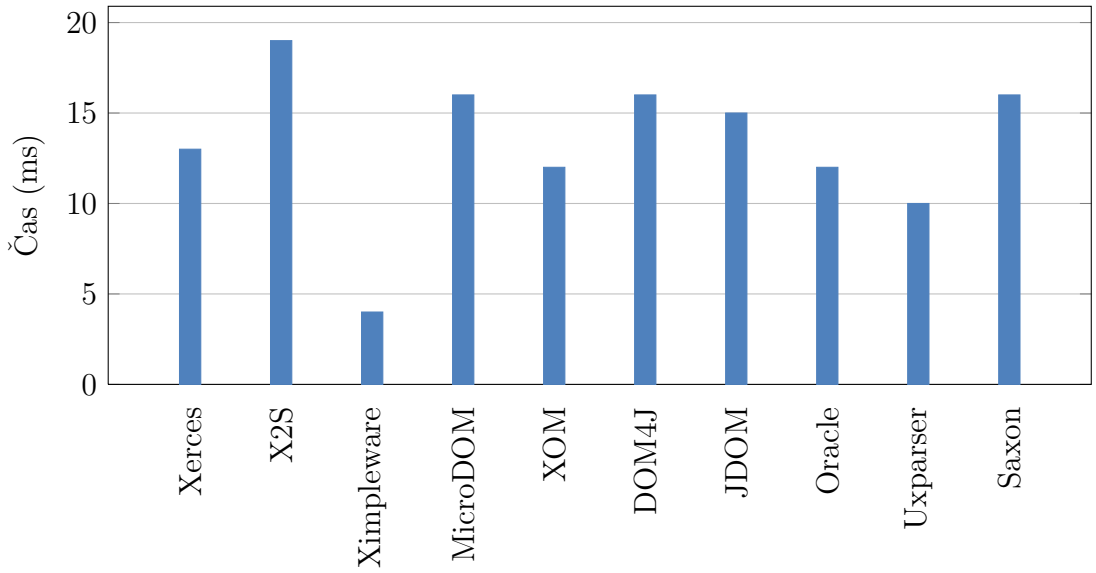
Při testování náročnosti parserů na operační paměť nejvíce vyčnívaly parsery MicroDOM a X2S, oba dva parsery potřebovaly více než 3MB paměti MicroDOM dokonce spotřeboval téměř 12MB paměti. Nejméně paměti zabraly parsery Saxon, OracleDOM a DOM4J, které shodně potřebovali 1 498kB paměti.

Zdaleka nejvyšší paměťovou propustnost měl Ximpleware parser dosáhl téměř rychlosti 52MB/s, většina ostatních parserů skončila v rozmezí cca 19 - 10 MB/s. S výjimkou Xerces a X2S parseru, které zejména kvůli dlouhé době parsování spadly na rychlosti 4 412 kB/s(Xerces) a 2 823 kB/s(X2S)

As You Like It

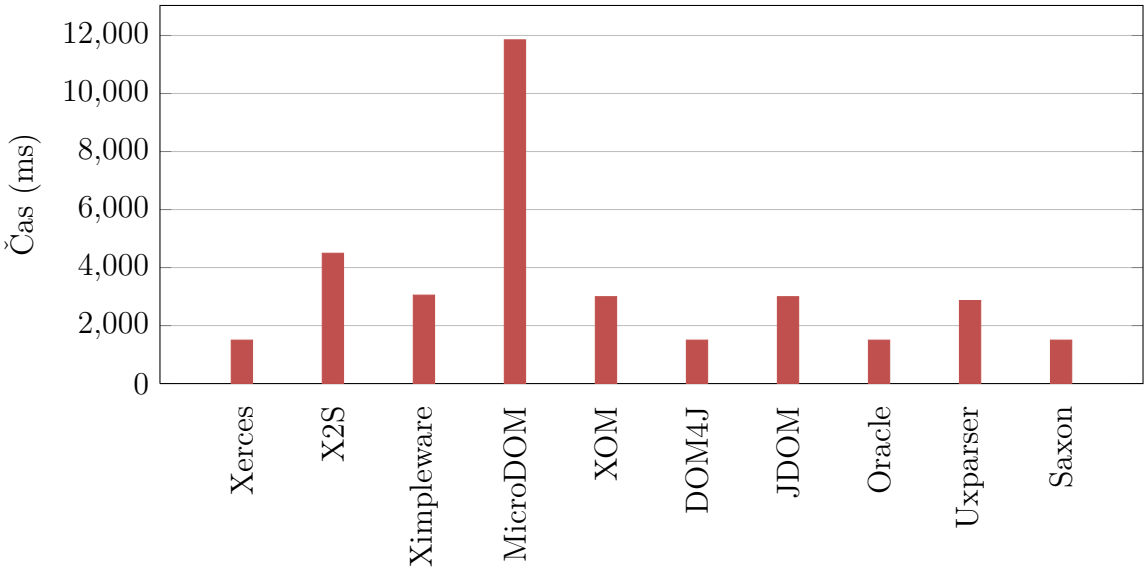
Jméno parseru	**Doba zpracování(ms)**	**Využitá paměť(kB)**	**BitRate (kB/s)**
Xerces	13	1 498	14 910
X2S	19	4 493	10 201
Ximpleware	4	3 049	48 457
MicroDOM	16	11 850	12 114
XOM	12	2 996	16 152
DOM4J	16	1 498	12 114
JDOM	17	2 996	11 402
Oracle DOM	12	1 498	16 152
Uxparser	10	2 863	19 383
Saxon	16	1 498	12 114

Tabulka 6.2: Hodnoty hry: As You Like It



Obrázek 6.3: As You Like It: Doba zpracování

V testu rychlosti jako jediný vybočil z řady Ximpleware parser který udržel rychlost 4ms, následován opět Uxparserem s 10ms, Xerces se s 13ms dostal z 9. na 5. místo jako poslední opět skončil X2S parser s časem 19ms.



Obrázek 6.4: As You Like It: Využitá paměť

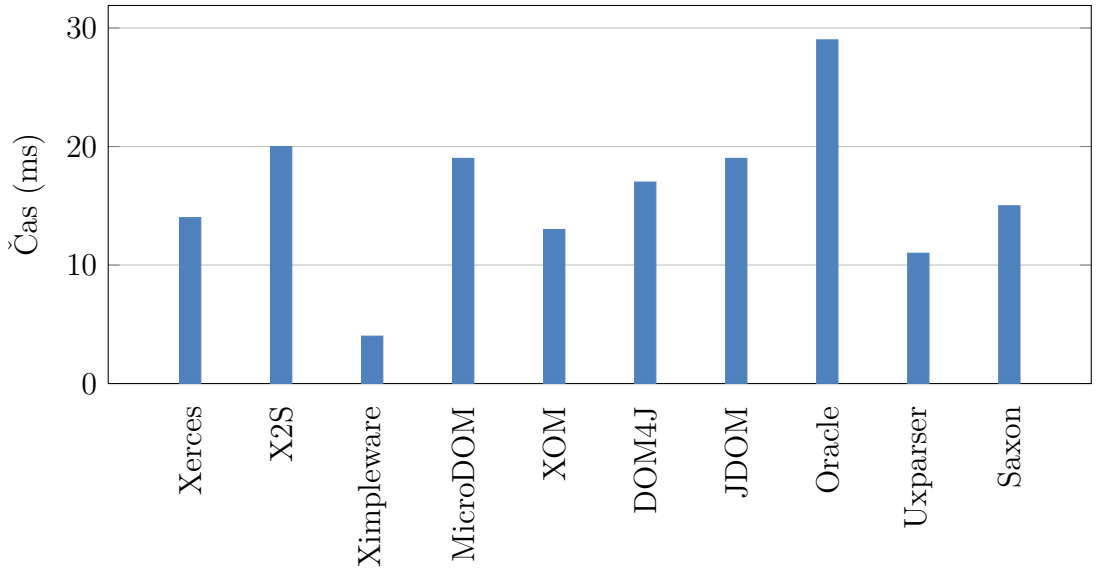
Hned 4 parsery skončily zároveň na spotřebě 1 498kB paměti a sice Saxon, Oracle DOM, DOM4J a Xerces. Parser MicroDOM opět použil výrazně nejvíce paměti celých 11 850kB. Druhým paměťově nejnáročnějším parserem byl parser X2S se spotřebou 4 493 kB paměti.

Nejvyšší propustnosti opět dosáhl parser Ximpleware s 48 457 kB/s na druhém místě se umístil Uxparser s 19 383kB/s a poslední místo obsadil X2S parser s 10 201kB/s.

Romeo and Juliet

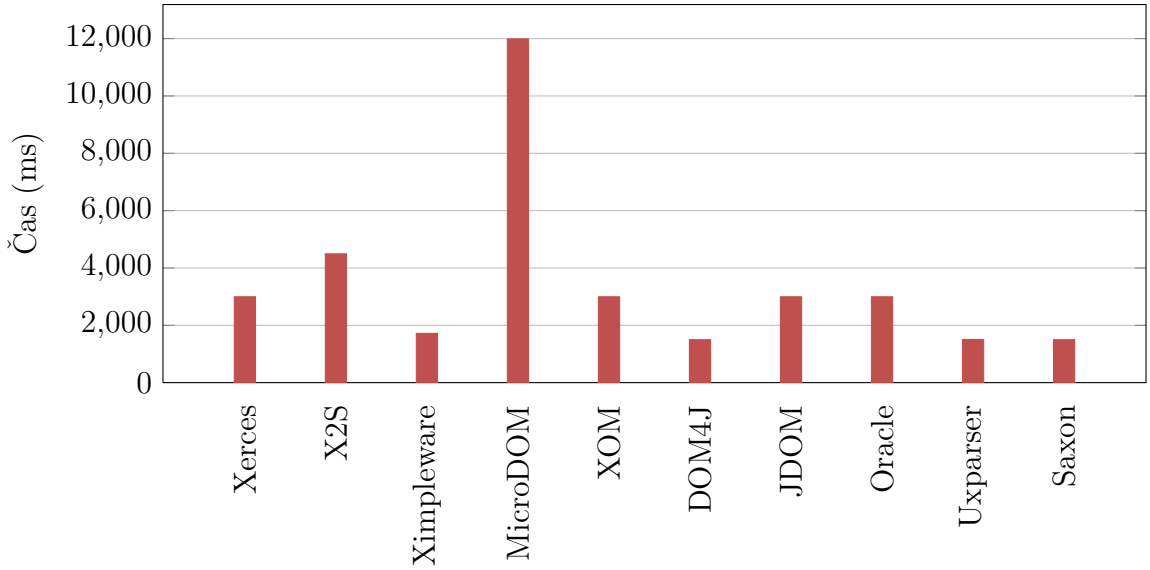
Jméno parseru	**Doba zpracování(ms)**	**Využitá paměť(kB)**	**BitRate (kB/s)**
Xerces	14	2 996	15 733
X2S	20	4 493	11 013
Ximpleware	4	1 718	55 065
MicroDOM	19	11 990	11 593
XOM	13	2 996	16 943
DOM4J	17	1 498	12 957
JDOM	19	2 996	11 593
Oracle DOM	29	2 995	7 595
Uxparser	11	1 506	20 024
Saxon	15	1 498	14 684

Tabulka 6.3: Hodnoty hry: Romeo and Juliet



Obrázek 6.5: Romeo and Juliet: Doba zpracování

Na konci rychlostního testu tentokrát skončil Oracle DOM parser s časem 29ms, parser X2S byl na 9. místě s 20ms. Ximpleware si stále udržuje rychlost 4ms opět následován Uxparserem s 11ms. O jedno místo v žebříčku se posunul i Xerces a dostal se na 4. místo.



Obrázek 6.6: Romeo and Juliet: Využitá paměť

V paměťovém testu se na poslední místo vrátil MicroDOM s 11 990kB, následován X2S parserem s 4 493kB. O první místo se v tom testu dělí Saxon a DOM4J s 1 498kB paměti následuje je Uxparser s 1 506kB.

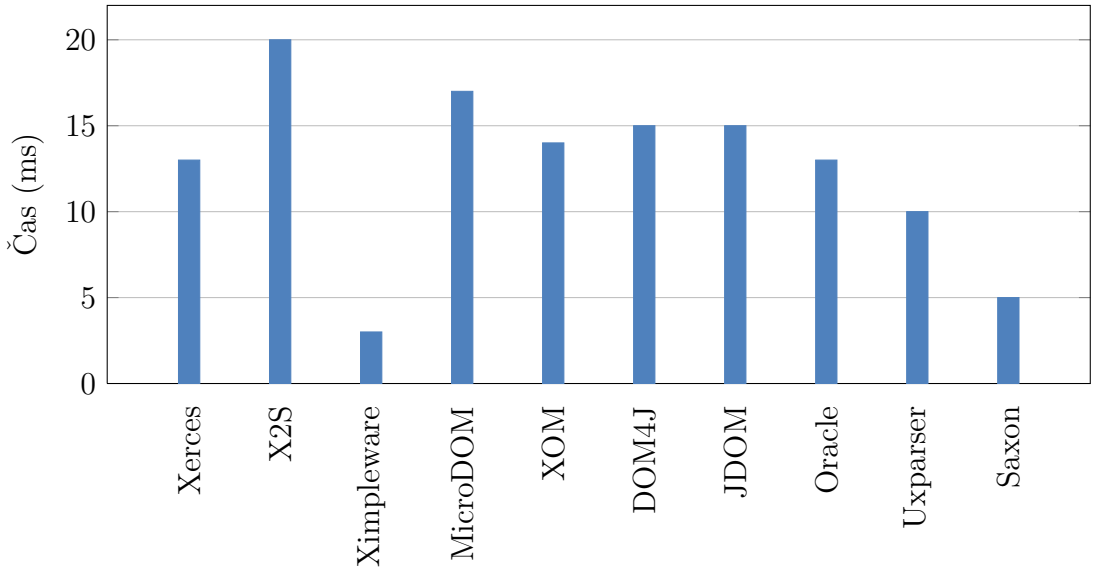
V testu přenosové rychlosti se propadl Oracle DOM parser jako jediný se nedostal nad 10 000 kB/s a dosáhl pouze rychlosti 7 595kB/s. Na druhé straně parsery Ximpleware

a Uxparser se dostali jako jediné dva nad hranici 20 000kB/s. Uxparser dosáhl rychlosti 20 024kB/s a Ximpleware parser dokonce rychlosti 55 065 kB/s, třetí nejrychlejší skončil XOM parser s 16 943 kB/s.

Twelfth Night

Jméno parseru	**Doba zpracování(ms)**	**Využitá paměť(kB)**	**BitRate (kB/s)**
Xerces	13	1 498	14 475
X2S	20	4 493	9 408
Ximpleware	3	1 686	62 723
MicroDOM	17	10 492	11 069
XOM	14	2 996	13 441
DOM4J	15	1 498	12 544
JDOM	15	2 996	12 545
Oracle DOM	13	1 498	14 475
Uxparser	10	1 506	18 817
Saxon	5	1 498	37 634

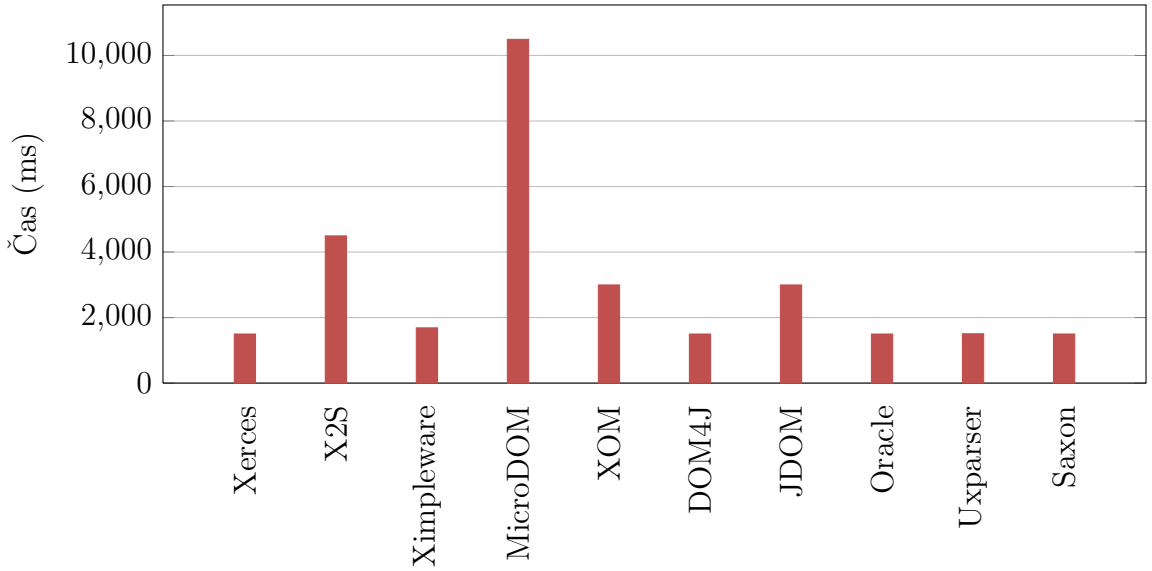
Tabulka 6.4: Hodnoty hry: Twelfth Night



Obrázek 6.7: Twelfth Night: Doba zpracování

V testu hry Twelfth Night Ximpleware překonal svůj dosavadní rekord a dostal se na čas 3ms. Na druhém místě vystřídal Uxparser s 10ms parser Saxon, který soubor

rozparsoval za 5ms. Nejpomalejším parserem byl opět parser X2S jako jediný se dostal na čas 20ms, druhým nejpomalejším parserem se stal MicroDOM s časem 17ms.



Obrázek 6.8: Twelfth Night: Využitá paměť

Většina parserů se v tomto testu dostala pod hranici 2 000kB. O nejmenší naměřenou hodnotu 1 498kB se dělí parsery Xerces, DOM4J, Oracle DOM a Saxon. Nad hranici 2 000kB se dostali parsery XOM a JDOM, které spotřebovaly shodně po 2 996kB následované X2S parserem s 4 493kB a poté s velkým odstupem paměťově nejnáročnější MicroDOM.

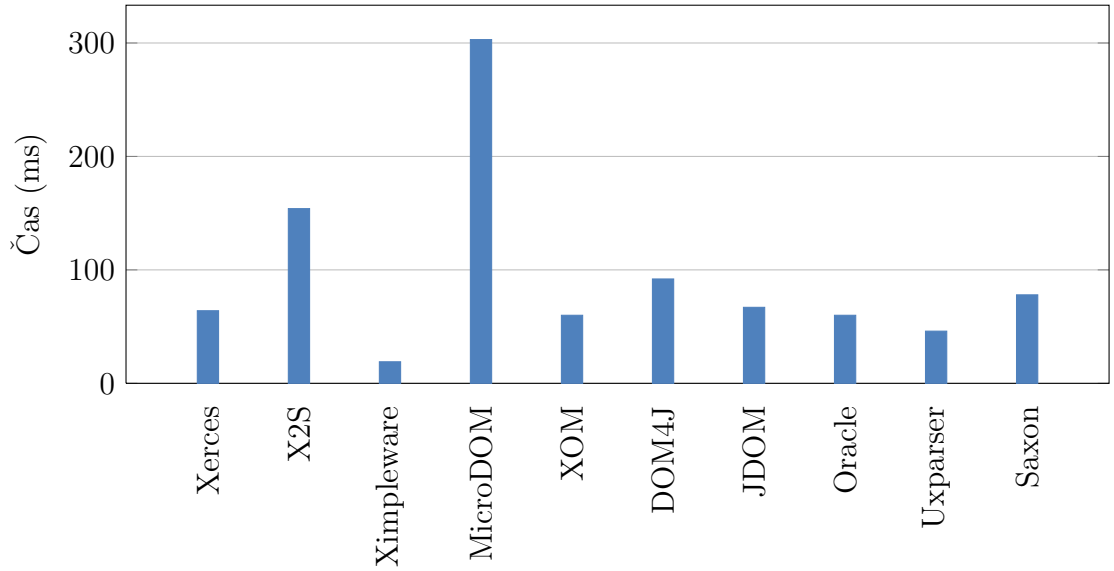
Nejvyšší přenosové rychlosti opět dosáhl Ximpleware parser s 62 723kB/s, nad 30MB/s se dostal ještě Saxon parser s 37 634kB/s. Nejpomalejším ze všech parserů a jediným parserem co se dostal pod hranici 10 000kB/s se stal X2S parser, který dosáhl rychlosti pouze 9 408kB/s.

Celkové zhodnocení: Z výsledků je patrná jasná převaha Ximpleware parseru při parsování malých souborů. Parser Ximpleware je prokazatelně nejrychlejší a má nejvyšší propustnost, propadl se pouze v nárocích na paměť. Naopak malé soubory nejdéle parsují parsery X2S a Xerces. Pomalejší než X2S parser byl pouze parser od Oraclu u souboru hry Romeo a Julie. Parser X2S má také poměrně vysoké nároky na operační paměť, z čehož vyplývá celkově nejnižší propustnost. Obecně nejvíce paměti spotřebovává parser MicroDOM.

6.1.2 Hry jako jedna kolekce

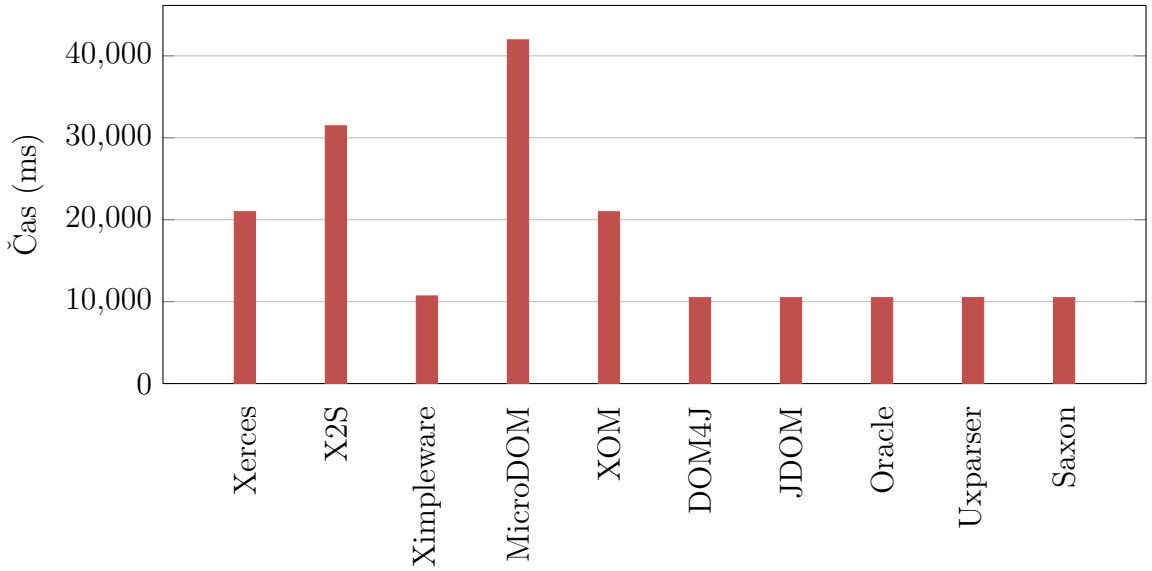
Jméno parseru	Doba zpracování(ms)	Využitá paměť(kB)	BitRate (kB/s)
Xerces	64	20 972	12 719
X2S	154	31 458	5 294
Ximpleware	19	10 698	42 843
MicroDOM	303	41 951	2 690
XOM	60	20 972	13 587
DOM4J	92	10 486	8 861
JDOM	67	10 486	12 167
Oracle DOM	60	10 486	13 587
Uxparser	46	10 494	17 722
Saxon	78	10 486	10 451

Tabulka 6.5: Naměřené výsledky kolekce her W. Shakespeara



Obrázek 6.9: Kolekce her W. Shakespeara: Doba zpracování

Také při parsování celé kolekce her najednou se dostáváme k podobným výsledkům jako u samostatných souborů. Největí změnou bylo, že MicroDOM(303ms) parser potřeboval na zpracování kolekce souborů téměř dvojnásobek času než nyní předposlední X2S parser, který kolekci zpracoval za 154ms. Všechny ostatní parsery se dostali pod čas 100ms. Tradičně nejrychlejším byl parser Ximpeware s časem 19ms, následovaný Uxparserem, který dokončil parsování za 46ms.



Obrázek 6.10: Kolekce her W. Shakespeara: Využitá paměť

V tomto testu většina parserů potřebovala pro práci okolo 10 500kB paměti. Čtyři parsery se dostaly nad hodnotu 20 000kB. Nejvíce paměti potřeboval MicroDOM parser a sice 41 951kB následvaný parserem X2S s 31 458kB. Parsery Xerces a XOM shodně spotřebovaly 20 972kB. Nejméně náročná čtveřice parserů DOM4J, JDOM, Oracle DOM a Saxon spotřebovala shodně 10 486kB paměti.

Nejvyšší přenosové rychlosti opět dosáhl parser Ximpleware a sice 42 843kB/s, na opačném konci spektra skončil parser MicroDOM s pouhými 2 690kB/s. Pod hranicí 10 000kB/s ještě zůstaly parsery X2S s rychlostí 5 294kB/s a DOM4J, který dosáhl rychlosti 8 861kB/s. Za zmínku stojí ještě druhý nejrychlejší Uxparser s přenosovou rychlostí 17 722kB/s

Celkové zhodnocení: Velmi podobný výsledek jako při testování jednotlivých souborů. Ximpleware je jasně nejrychlejší s nejvyšší propustností. Test ukázal, že nejnevhodnější na testování kolekcí malých souborů je parser MicroDOM, který byl v testu nejpomalejší a spotřeboval nejvíce paměti, zatímo většina parserů potřebovala ke zpracování souborů okolo 1MB paměti parser MicroDOM použil 4MB.

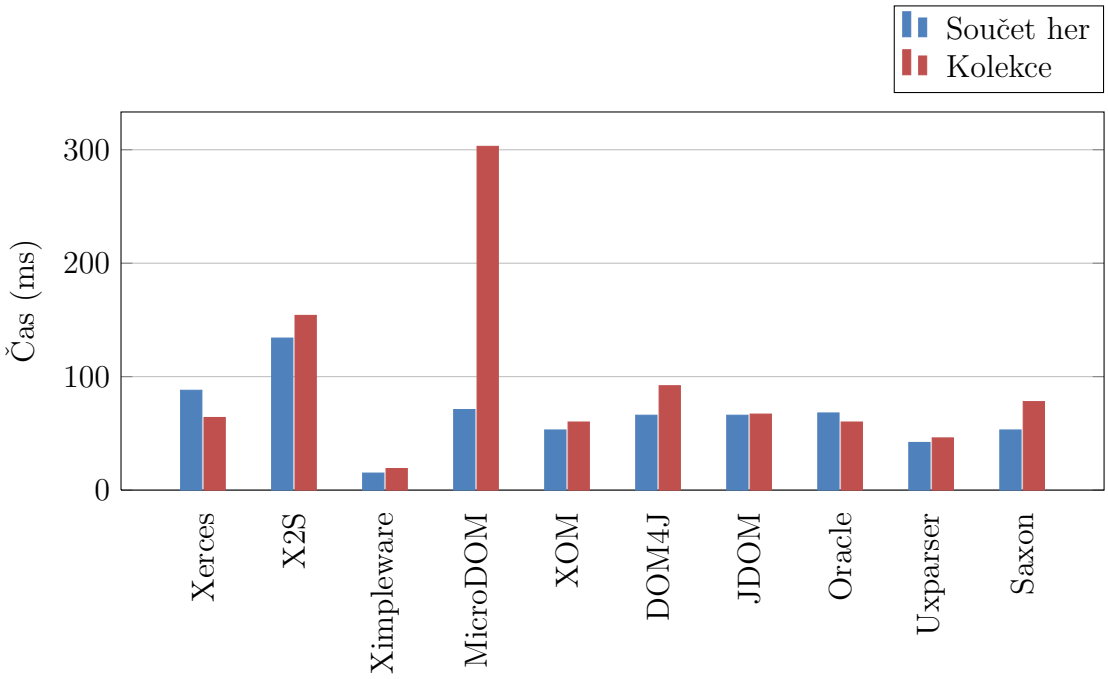
6.1.3 Porovnání součtu výsledků jednotlivých souborů a meření kolekce souborů

Cíl porovnávání: Cílem tohoto testu bylo zjistit rozdíl mezi rozparsováním souborů po jednom a součtu výsledků (propustnost byla dělena čtyřma), nebo jestli je výhodnější parsovat celou kolekci.

Toto porovnávání vychází z tabulek 6.5 a 6.6

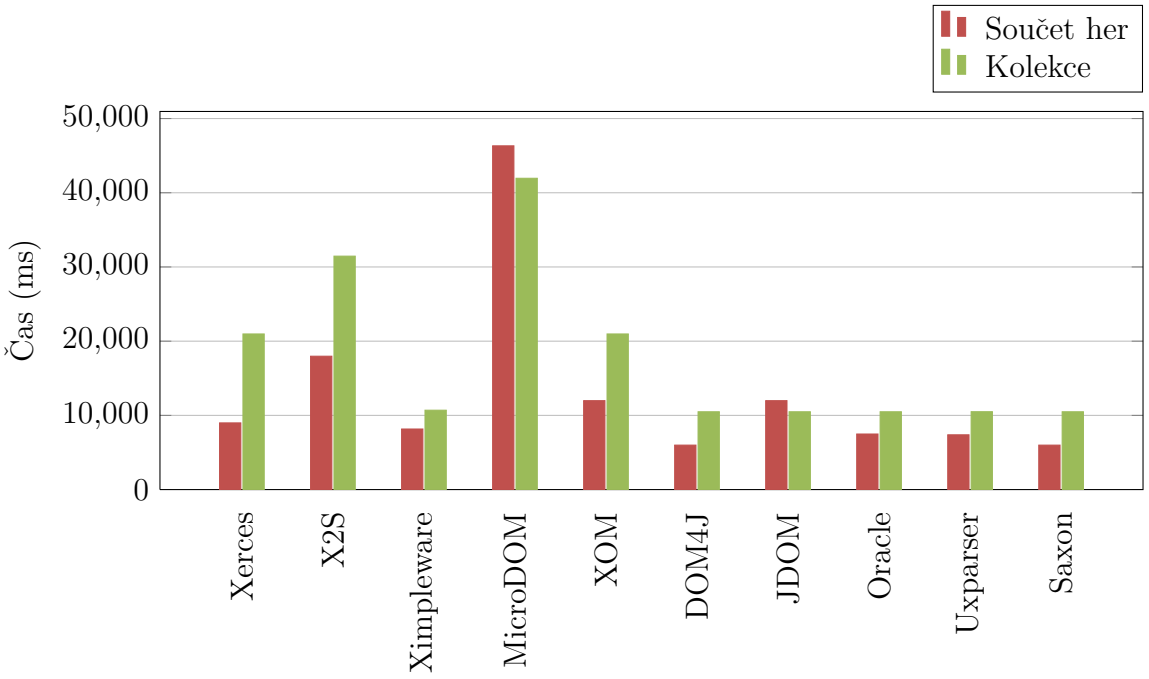
Jméno parseru	**Doba zpracování(ms)**	**Využitá paměť(kB)**	**BitRate (kB/s)**
Xerces	88	8 986	12 382
X2S	134	17 973	8 362
Ximpleware	15	8 163	54 796
MicroDOM	71	46 321	11 480
XOM	53	11 982	15 415
DOM4J	66	5 992	12 345
JDOM	66	11 982	12 414
Oracle DOM	68	7 489	13 337
Uxparser	42	7 381	19 369
Saxon	53	5 992	19 222

Tabulka 6.6: Součet měření jednotlivých souborů her W. S.



Obrázek 6.11: Porovnání testů her W. Shakespeara: Doba zpracování

Na celém grafu je jistě nejvýraznější rozdíl u časů kterých dosáhl MicroDOM parser, kde součet časů v dosáhl hodnoty 71ms, zatímco parsování celé kolekce zabralo 303ms, rozdíl mezi oběma hodnotami je tedy velmi vysoký rozdíl celých 232ms. U žádného z dalších parserů tak vysoký rozdíl mezi naměřenými hodnotami není, nejmenšího rozdílu dosáhl parser JDOM, kde byl rozdíl pouhou 1ms ve prospěch testování celé kolekce.



Obrázek 6.12: Porovnání testů her W. Shakespeara:: Využitá paměť

Také v tom případě nastal výrazný rozdíl mezi celkovými paměťovými nároky u jednotlivých her a celé kolekce. Nesrovnalost se v tomto případě projevila nejvíce u parseru Xerces, který potřeboval na vyparsování celé kolekce 20 972kB paměti, zatímco na součet her mu stačilo pouze 8 986kB, rozdíl je tedy celých 11 986kB, nejmenší rozdíl nastal opět u parseru JDOM, který v součtu potřeboval 11 982kB a pro celou kolekci 10 486kB, rozdíl byl tedy 1 496kB, což je přibližně desekrát méně neuž u Xercesu.

Celkové zhodnocení: Co do rychlosti u většiny parserů je mírně vyšší čas při parsování celé kolekce, výjimkou je parser MicroDOM, kde parsování celé kolekce trvá výrazně déle, než parsování jestlivých souborů a parsery Xerces a Oracle, kde je naopak rychlejší parsování celé kolekce.

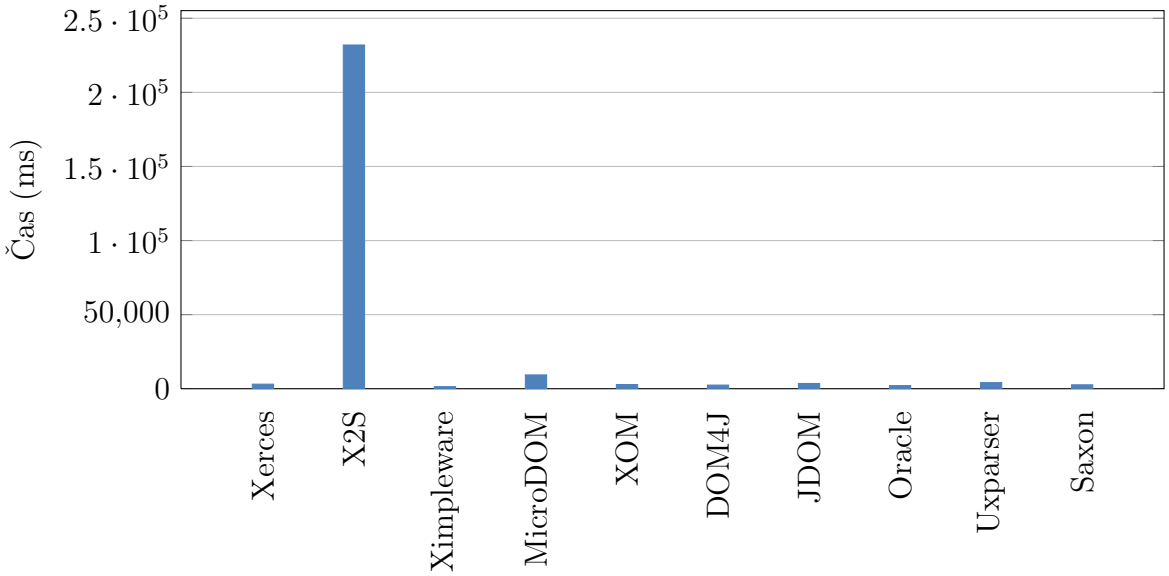
Ve spotřebě paměti se opět ukázalo, že více paměti potřebuje při testování celé kolekce většina parserů. Tento rozdíl je poměrně výrazný u parserů Xerces a X2S. Parsery MicroDOM a JDOM naopak spotřebovaly méně paměti při parsování kolekce.

Jak lze vidět z grafu propustnosti a naznačily předchozí testy, většina grafů má nižší propustnost při testování celé kolekce. Nejmarkantnější je tento rozdíl u parserů Ximpleware, MicroDOM a Saxon. Parsery Xerces a Oracle jsou mají mírně vyšší propustnost při testování celé kolekce.

6.2 EPA Geospatial Data Access Project

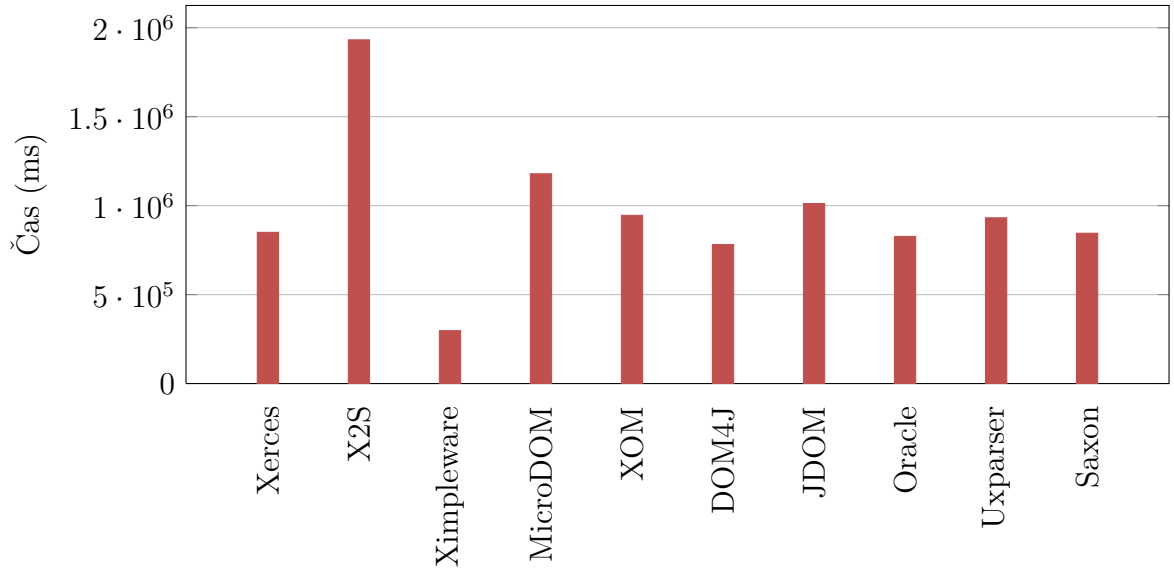
Jméno parseru	Doba zpracování(ms)	Využitá paměť(kB)	BitRate (kB/s)
Xerces	3 085	849 979	73 842
X2S	231 961	1 931 908	982
Ximpleware	1 409	297 235	161 676
MicroDOM	9 328	1 179 029	24 421
XOM	2 837	945 532	80 297
DOM4J	2 464	781 081	92 452
JDOM	3 537	1 011 705	64 405
Oracle DOM	2 130	826 962	106 949
Uxparser	4 122	931 659	55 265
Saxon	2 657	844 943	85 736

Tabulka 6.7: Výsledky měření: EPA



Obrázek 6.13: EPA: Doba zpracování

Na tomto testu je nejvýraznější čas který potřeboval na zpracování souboru X2S parser, zatímco většina parserů skončila během 3 vteřin, potřeboval parser X2S téměř 232 vteřin, tedy skoro 4 minuty. Druhým nejpomalejším parserem byl MicroDOM parser, který potřeboval na prací pouze 9 vteřin 328ms. Nejrychlejší byl opět Ximpleware parser s časem 1 409ms následovaná Oracle DOM parserem, který potřeboval 2 130ms.



Obrázek 6.14: EPA: Celkový přehled

Parser X2S se výrazně odlišil i v nárocích na paměť na zpracování geoprostorých dat potřeboval 1 931 908kB, což je o celých 752 879kB více než kolik potřeboval druhý nejnáročnější parser MicroDOM, který potřeboval 1 179 029kB paměti, nad 1 000 000kB paměti potřeboval ještě JDOM parser se spotřebou 1 011 705kB, všechny ostatní parsery se již vešly pod tuto hranici. Nejměnší paměťovou spotřebu měl parser Ximpleware s 297 235kB následovaný parserem DOM4J, který potřeboval 781 081kB místa v paměti.

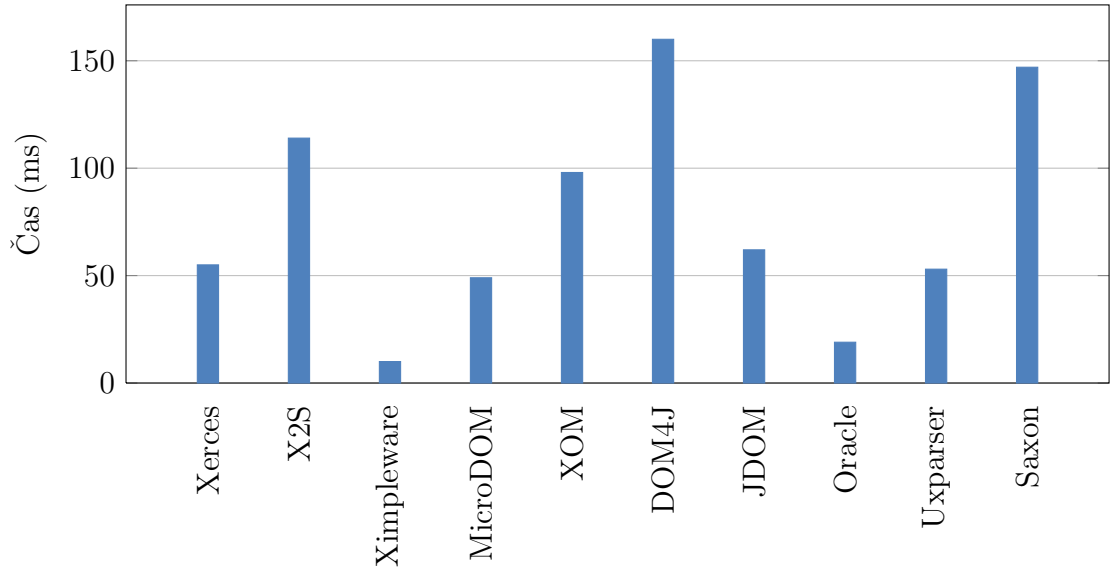
Jak vyplývá z tabulky 6.7 a předchozích měření měl X2S parser také nejhorší propustnosti pouhých 982kB/s, druhý nejhorší parser MicroDOM měl propustnost 24 421kB/s, což je o 23 439kB/s více. Hranici 100MB/s překonaly pouze dva parsery a sice Ximpleware s rychlostí 161 676kB/s a Oracle DOM parser s rychlostí 106 949kB/s

Celkové zhodnocení: V tomto testu byl rozhodně nejpomalejší parser X2S s časem 231 961ms(3min 52 vteřin) za druhý nejpomalejší parser byl MicroDOM s 9 328ms. Parser X2S měl také nejvyšší nároky na paměť(1,9GB) druhý byl MicroDOM, který potřeboval 1,78GB paměti. Nejrychlejší a zároveň nejméně pamětově náročný byl parser Ximpleware(1 409ms a 297MB) z čehož také vyplývá, že Ximpleware měl také nejvyšší propustnost.

Jméno parseru	Doba zpracování(ms)	Využitá paměť(kB)	BitRate (kB/s)
Xerces	55	2 995	8 566
X2S	114	5 991	4 133
Ximpleware	10	1 969	47 112
MicroDOM	49	23 963	9 617
XOM	98	2 996	4 807
DOM4J	160	2 996	2 945
JDOM	62	2 996	7 599
Oracle DOM	19	2 996	24 796
Uxparser	53	3 004	8 889
Saxon	147	3 124	3 205

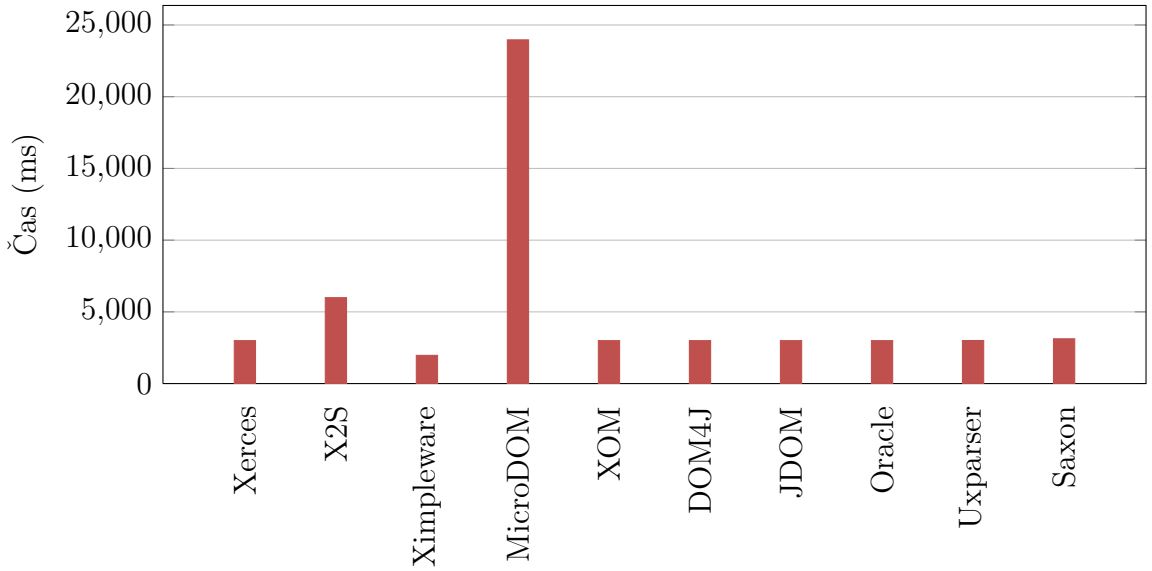
Tabulka 6.8: Výsledky měření: FDCCI

6.3 FDCCI



Obrázek 6.15: FDCCI: Doba zpracování

Zpracování tohoto souboru zabralo nejdéle parseru DOM4J, který potřeboval 160ms, druhým nejpomalejším parserem byl Saxon se 147ms. X2S parser s časem 114ms byl třetím nejpomalejším parserem. Nejrychlejším parserem byl tradičně Ximpleware s časem 10ms, následovaný Oracle DOM parserem s časem 19ms, tradičně rychlý Uxparser skončil s časem 53ms 4. za MicroDOM parserem s časem 49ms.



Obrázek 6.16: FDCCI: Celkový přehled

Paměťově nejnáročnějším parserem byl opět MicroDOM parser, který potřeboval 23 963kB paměti, druhý nejnáročnější parser potřeboval o necelých 18 000kB paměti méně a byl jím parser X2S s 5 991kB paměti. Většina parserů potřebovala pro práci okolo 3 000kBpaměti. Nejmenší spotřebu měl parser Ximpleware s 1 969kB následovaný parserem Xerces který potřeboval 2 995kB paměti.

Parsery s nejvyššími přenosovými rychlostmi se staly Ximpleware s rychlostí 47 112kB/s a Oracle DOM s 24 796kB/s, toto jsou také jediné dva parsery které se dostaly nad hodnotu 10MB/s. Parserem s nejnižší přenosovou rychlostí se stal DOM4J parser, ten dosáhl rychlosti 2 945kB/s následovaný Saxon parserem s rychlostí 3 205kB/s.

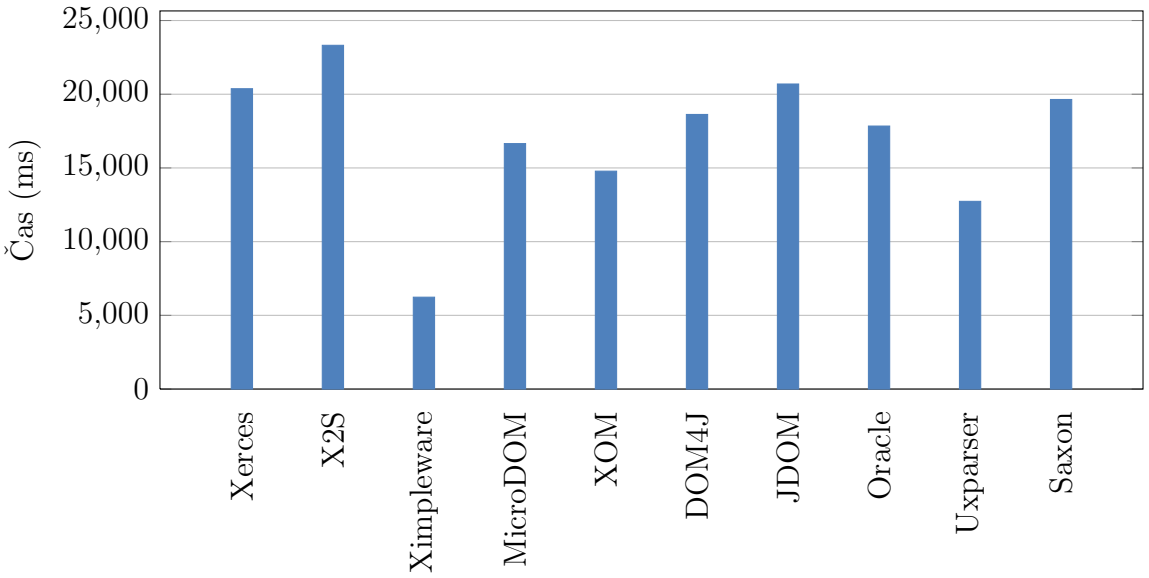
Celkové zhodnocení: I u tohoto testu byl Ximpleware nejrychleší a potřeboval nejméně paměti. Nejdéle pracoval parser DOM4J a parser Saxon, rozdíl mezi nejrychlejším a nejpomalějším časem je 150ms. Výrazně nejvíce paměti opět potřeboval parser MicroDOM(23,96MB oproti 1,96MB u Ximpleware).

6.4 Inex-Wiki

6.4.1 Měření kolekce 10 000 souborů

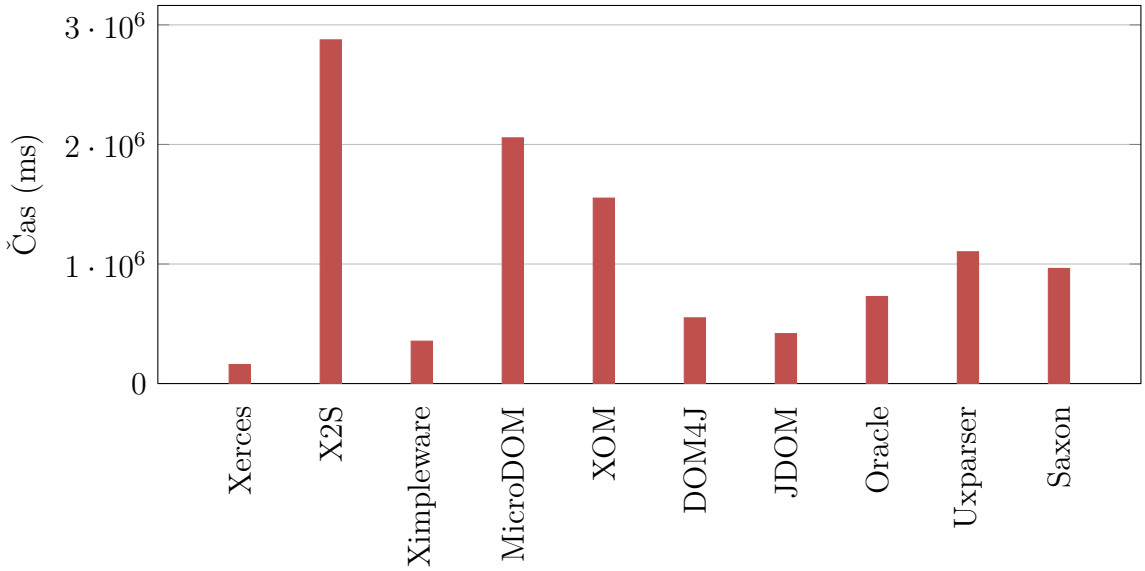
Jméno parseru	Doba zpracování(ms)	Využitá paměť(kB)	BitRate (kB/s)
Xerces	20 374	159 717	7 110
X2S	23 319	2 874 863	6 216
Ximpleware	6 232	354 926	23 246
MicroDOM	16 651	2 055 187	8 706
XOM	14 781	1 551 276	9 807
DOM4J	18 628	549 952	7 782
JDOM	20 694	418 108	7 007
Oracle DOM	17 841	728 481	8 125
Uxparser	12 733	1 102 414	11 384
Saxon	19 644	962 999	7 379

Tabulka 6.9: Výsledky měření: Inex-Wiki 10 000 souborů



Obrázek 6.17: Inex-Wiki 10k: Doba zpracování

V tomto testu se jediný Ximpleware parser s časem 6 232ms dostal pod 10 vteřin. Uxparser, který byl druhý nejrychlejší práci dokončil v čase 12 733ms. Pod hranici 15 vteřin se dostal ještě XOM parser s časem 14 781ms. Tři nejpomalejší parsery potřebovaly na zpracování dokumentu více než 20 vteřin, byly to parsery X2S s časem 23 319ms, JDOM skončil v čase 20 694ms a Xerces s časem 20 347ms.



Obrázek 6.18: Inex-Wiki 10k: Využitá paměť

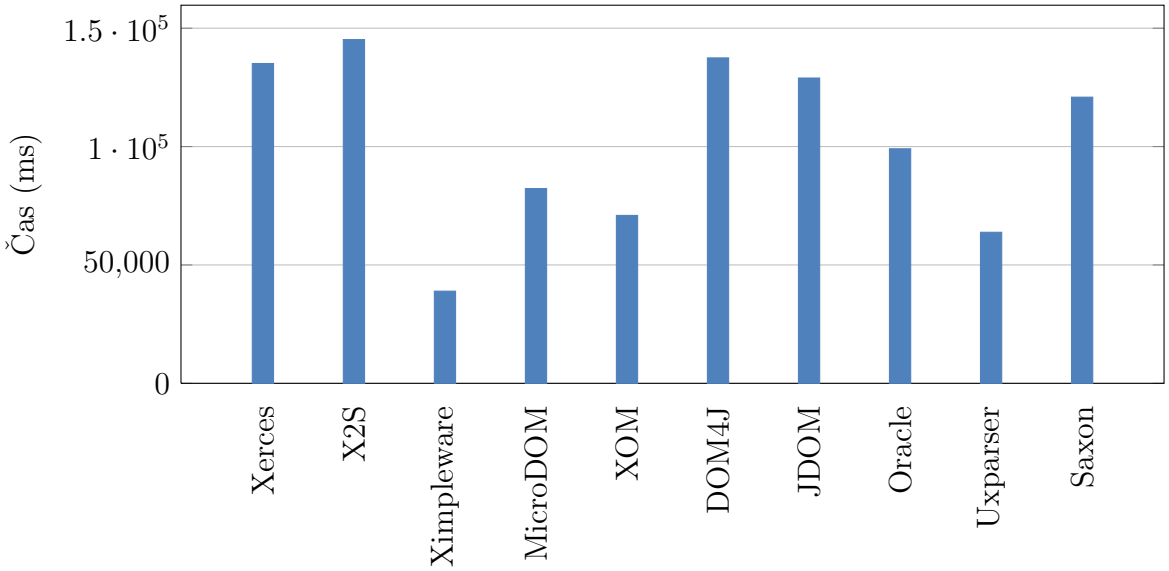
Parser X2S byl opět i nejvíce paměťově náročným parserem na práci potřeboval 2 874 863kB paměti, daleko za ním následoval MicroDOM parser s 2 055 187kB paměti. Nad 1GB paměti potřebovali ještě parsery XOM(1 551 276kB) a Uxparser(1 102 414kB). Nejméně paměťově náročným parserem byl Xerces se 159 717kB za ním následoval se spotřebou 354 926kB Ximpleware a na třetím místě skončil JDOM, který potřeboval 418 108kB paměti. Dvojice parserů s nejvyšší propustností byla Ximpleware s 23 246kB/s a Uxparser s 11 384kB/s. Na konci žebříčku skončil X2S parser s rychlostí 6 216kB/s následovaný JDOM parserem přenosovou rychlostí 7 007kB/s.

Celkové zhodnocení: I v tomto testu byl opět nejrychlejší a měl také nejvyšší propustnost parser Ximpleware, nejméně paměti potřeboval pro změnu parser Xerces. Nejpomalější byl opět parser X2S, který tentokrát potřeboval i nejvíce paměti. Více než dvojnásobnou propustnost měl parser Ximpleware(23,24MB/s) oproti Uxparseru(11,38MB), který skončil druhý.

6.4.2 Měření kolekce 60 000 souborů

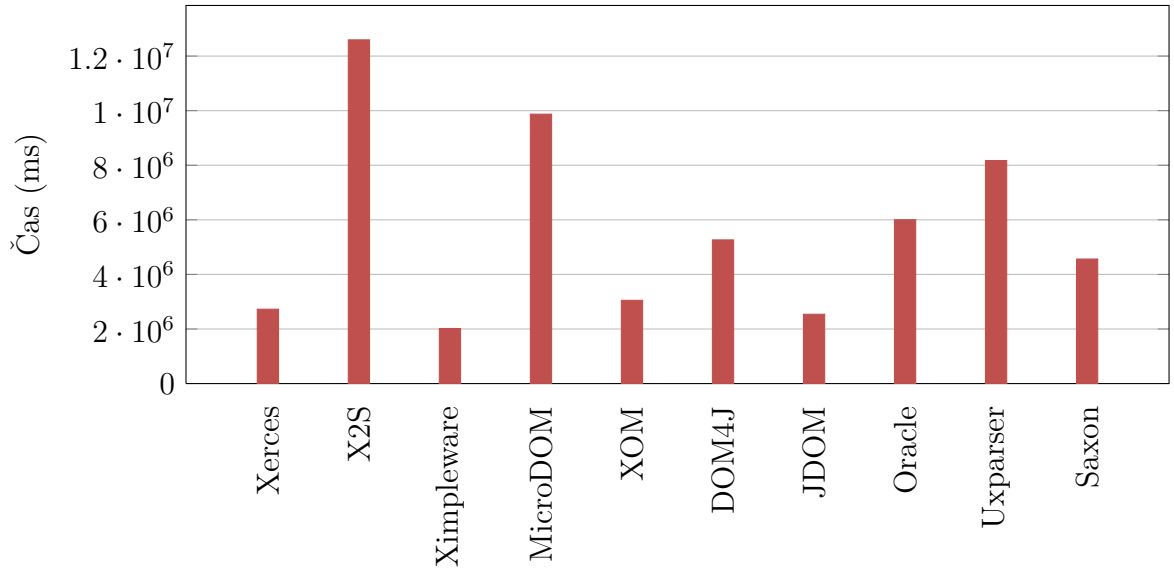
Jméno parseru	Doba zpracování(ms)	Využitá paměť(kB)	BitRate (kB/s)
Xerces	135 090	2 726 267	6 087
X2S	145 192	12 594 539	5 669
Ximpleware	38 935	2 018 857	21 129
MicroDOM	82 283	9 871 529	10 003
XOM	70 927	3 051 310	11 605
DOM4J	137 458	5 265 075	5 988
JDOM	128 961	2 540 416	6 383
Oracle DOM	99 070	6 002 595	8 308
Uxparser	63 809	8 172 937	12 899
Saxon	120 877	4 563 836	6 809

Tabulka 6.10: Výsledky měření: Inex-Wiki 60 000 souborů



Obrázek 6.19: Inex-Wiki 60k: Doba zpracování

Kolekci o 60 000 souborech nejrychleji zpracoval Ximpleware parser v čase 38 935ms, byl také jediným parserem, který soubor zpracoval pod 1 minutu. Za Ximplewarem následoval Uxparser s časem 63 809ms a XOM parser s časem 70 927ms. Nejdéle parsoval X2S parser, který potřeboval 145 192ms za ním následoval DOM4J s časem 137 458ms.



Obrázek 6.20: Inex-Wiki 60k: Využitá paměť

Nejméně paměti spotřeboval Ximpleware parser, který potřeboval 2GB paměti, za ním následoval se spotřebou 2,5GB JDOM parser. O 10GB paměti více potřeboval parser X2S, za ním následoval s necelými 10GB paměti MicroDOM parser.

V tomto testu se nad hranici rychlosti 10 000kB/s dostali pouze 4 parsery a sice Ximpleware parser s rychlostí 21 129kB/s, Uxparser dosáhl rychlosti 12 899kB/s a nakonec XOM a MicroDOM parser s rychlostmi 11 605 a 10 003kB/s

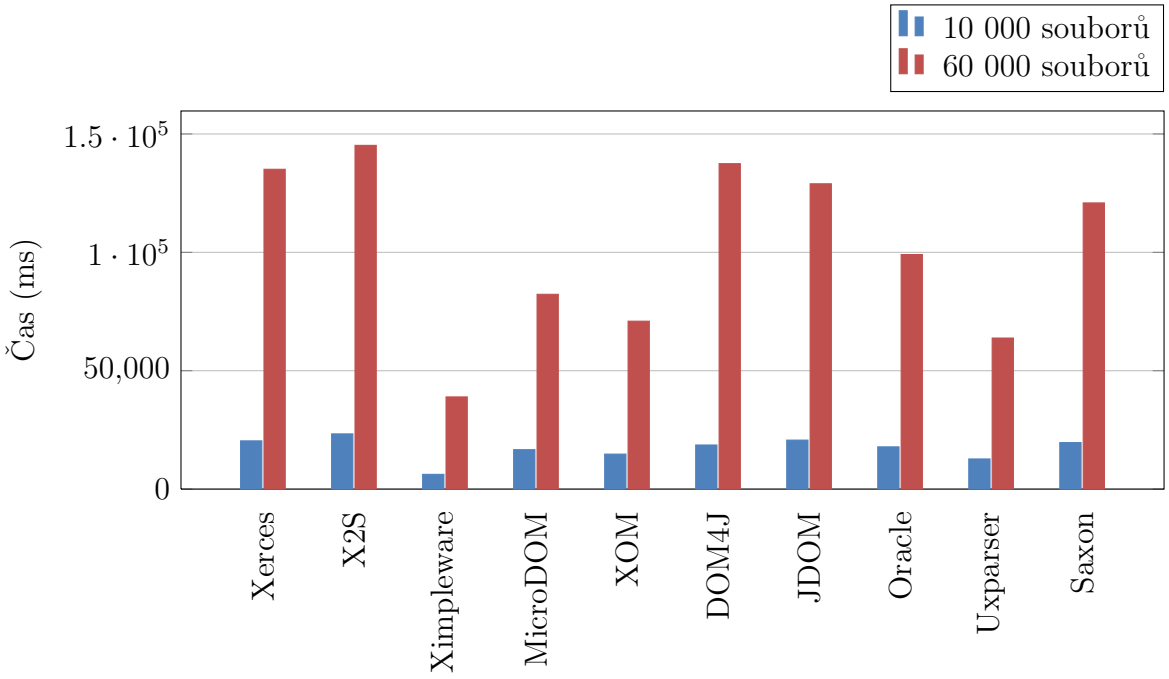
Celkové zhodnocení: V tomto testu se parser Ximpleware opet vrátil do čela ve všech sledovaných kategoriích. Parser X2S potřeboval pro práci nejvíce paměti a času: 2min 25 vteřin a 12,59GB, zatímco parser Ximplevare potřeboval: 38,94 vteřiny a 2GB paměti. Parser Ximpleware měl i poměrně velký náskok v propustnosti, mezi ním a druhým Uxparserem byl rozdíl 8,22MB/s.

6.4.3 Porovnání výsledků mezi 10 000 a 60 000 soubory

Cíl porovnánávní Cílem tohoto měření bylo zjisit jak se změní pozorované parametry. Přepokladem bylo, že by spotřeba paměti a doba zpracování by měla být přibližně šestinásobná, zatímco přenosová rychlot by se neměla výrazněji měnit.

Jméno parseru	10 000 souborů	60 000 souborů	Podíl(60k/10k)
XOM	14 781	70 927	4,799
MicroDOM	16 651	82 283	4,942
Uxparser	12 733	63 809	5,011
Oracle DOM	17 841	99 070	5,553
Saxon	19 644	120 877	6,153
X2S	23 319	145 192	6,226
JDOM	20 694	128 961	6,232
Ximpleware	6 232	38 935	6,248
Xerces	20 374	135 090	6,630
DOM4J	18 628	137 458	7,379

Tabulka 6.11: Inex-Wiki: Porovnání doby zpracování

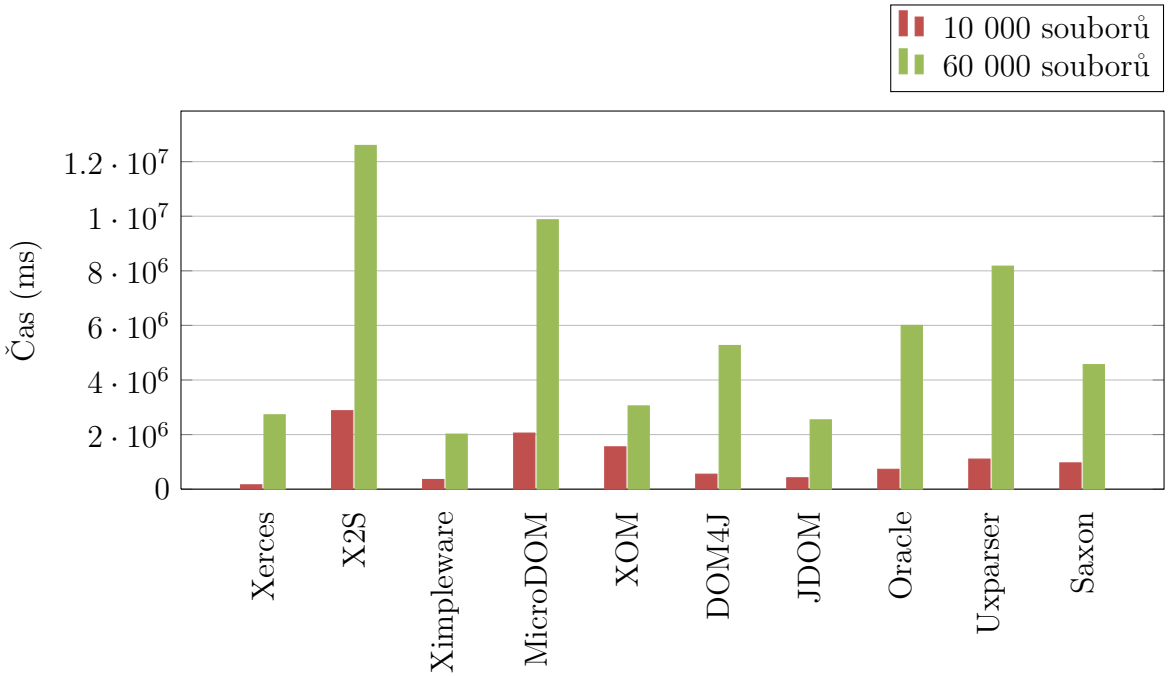


Obrázek 6.21: Porovnání Inex-Wiki: Doba zpracování

V tomto měření jsme zjistili, že nejmenší rozdíl mezi dobou zpracování 10 000 a 60 000 souborů je u XOM parseru, kde doba zpracování narostla přibližně 4,8krát následoval MicroDOM s 4,9 násobným nárůstem. Nejblíže předpokládanému šestinásobnému nárůstu doby zpracování se přiblížily parsery Saxon, X2S, JDOM a Ximpleware. Největší nárůst doby zpracování byl zaznamenán u parseru DOM4J, který potřeboval více než sedminásobek času pro zpracování kolekce o 10 000 souborech.

Jméno parseru	10 000 souborů	60 000 souborů	Podíl(60k/10k)
XOM	1 551 276	3 051 310	1,967
X2S	2 874 863	12 594 539	4,381
Saxon	962 999	4 563 836	4,739
MicroDOM	2 055 187	9 871 527	4,803
Ximpleware	354 926	2 018 857	5,688
JDOM	418 108	2 540 416	6,076
Uxparser	1 102 414	8 172 937	7,414
Oracle DOM	728 481	6 002 595	8,24
DOM4J	549 952	5 265 075	9,574
Xerces	159 717	2 726 267	17,069

Tabulka 6.12: Inex-Wiki: Porovnání využití paměti



Obrázek 6.22: Porovnání Inex-Wiki: Využitá paměť

Předpokládanému šestinásobku se nejvíce přiblížily parsery JDOM a Ximpleware, nejmenší nárust potřebné paměti byl zazanamenán u XOM parseru který nepotřeboval ani dvojnásobek místa, zatímco parser Xerces potřeboval více než 17krát více místa v paměti pro zpracování 60 000 souborů než u kolekce o 10 000 souborech.

Test porovnání propustnosti vyšel nejblíže předpokladům, největší odchylka - 33 setin nastala u parseru DOM4J, zcela nejblíže předpokládanému výsledku byl Oracle DOM parser, který se v podílu výsledků kolekcí odchýlil o 23 tisícin.

Celkové zhodnocení: Během měření nastaly od předpokládaného šestinásobku odchylky, ty mohly být mimo jiné způsobeny odlišnou průměrnou velikostí testovaných sou-

Jméno parseru	10 000 souborů	60 000 souborů	Podíl(60k/10k)
DOM4J	7 782	5 988	0,77
Xerces	7 110	6 087	0,856
Ximpleware	23 246	21 121	0,909
JDOM	7 005	6 383	0,911
X2S	6 216	5 669	0,912
Saxon	7 379	6 809	0,923
Oracle DOM	8 125	8 308	1,023
Uxparser	11 384	12 899	1,133
MicroDOM	8 706	10 003	1,149
XOM	9 807	11 605	1,183

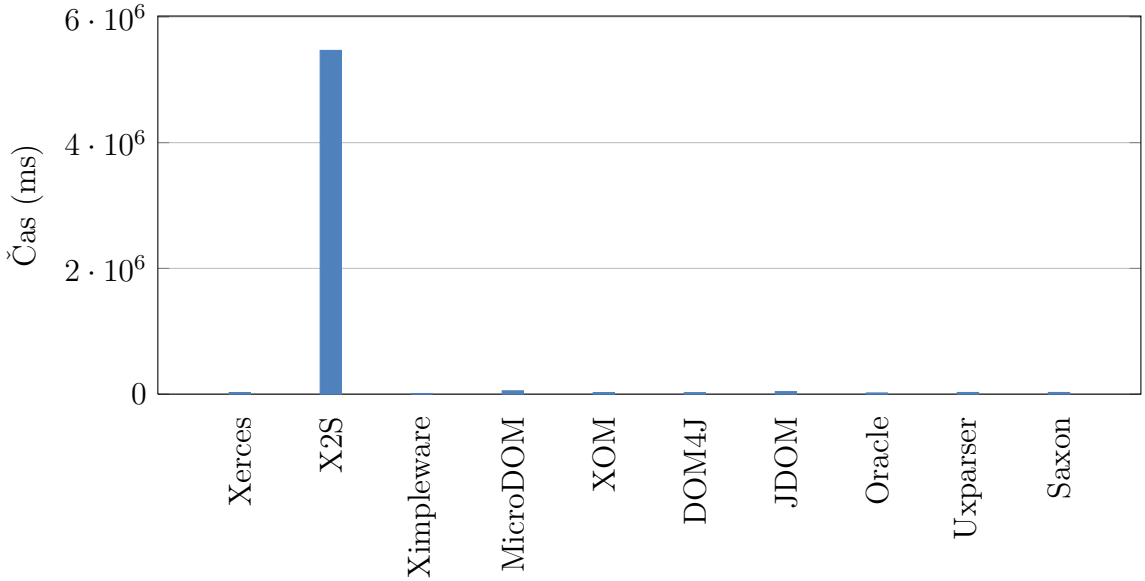
Tabulka 6.13: Inex-Wiki: Porovnání propustnosti

borů. Efekt na výsledek mohly mít také různé spouštěcí časy jednotlivých parserů. Největší odchylky od předpokladu nastaly u měření spotřebované paměti. XOM parser potřebný objem paměti zvětšil pouze 1,97krát, zatímco Xerces potřeboval 17krát více paměti. Z hlediska doby parsování se nejvíce směrem nahoru odchýlil parser DOM4J, kde se doba parsování zvýšila více než sedminásobně, na druhhou stranu doba parsování se u XOM parseru zvýšila pouze 4,8krát. Přenosová rychlost se nejvíce přiblížila předpokladům, vezmeme-li opět v potaz extrémy pak podíl rychlostí činil 0,77 u DOM4J a u XOM parseru 1,183.

6.5 PIR-PSD

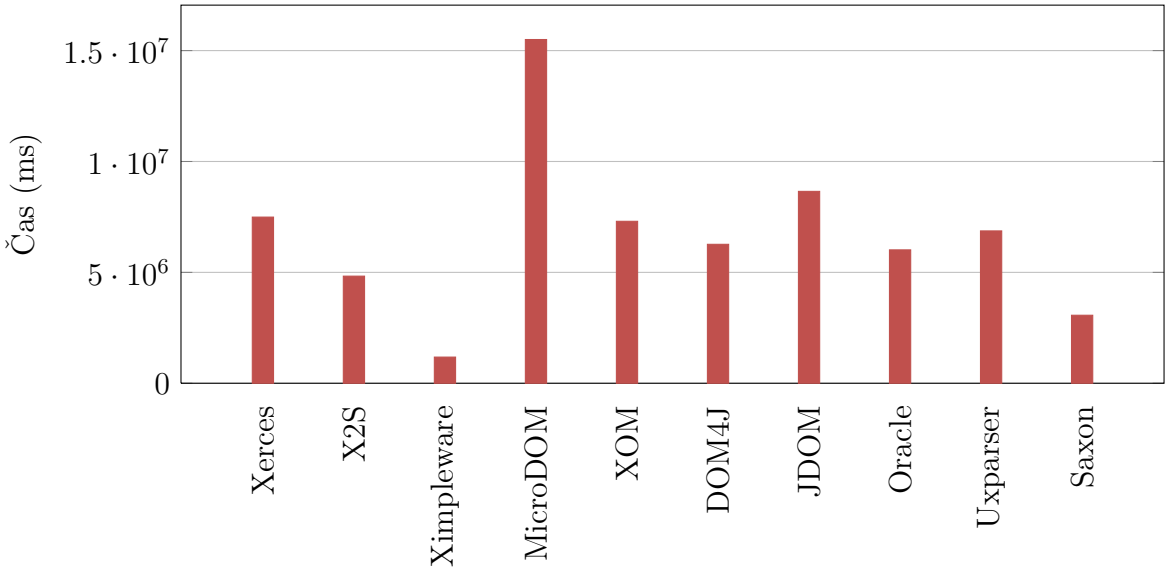
Jméno parseru	Doba zpracování(ms)	Využitá paměť(kB)	BitRate (kB/s)
Xerces	25 505	7 492 572	27 448
X2S	5 467 228	4 830 071	128
Ximpleware	10 499	1 179 754	66 678
MicroDOM	53 056	15 500 000	13 195
XOM	25 358	7 303 232	27 607
DOM4J	24 327	6 265 501	28 777
JDOM	40 240	8 652 552	17 397
Oracle DOM	19 273	6 014 761	36 323
Uxparser	26 489	6 869 561	26 428
Saxon	26 708	3 067 064	26 211

Tabulka 6.14: Výsledky měření: PIR-PSD



Obrázek 6.23: PIR-PSD: Doba zpracování

V testu mezinárodní databáze proteinů byl opět nejrychlejší Ximpleware parser s časem 10 499ms, druhý se umístil Oracle DOM s časem 19 273ms. Na opačném konci žebříčku skončil s enormním propadem parser X2S, který potřeboval 5 467 228ms(8minut, 52vteřin). Parser MicroDOM, který byl druhý nejpomalejší dokončil parsování za 53 vteřin, tedy téměř 8minut dříve.



Obrázek 6.24: PIR-PSD: Využitá paměť

S velkým náskokem skončil jako paměťově nejnáročnější parser MicroDOM se spotřebou 15,5GB paměti za ním následoval se spotřebou 8,6GB JDOM. Nejméně paměti potřeboval pro práci parser Ximpleware, který potřeboval pouze 1,1GB paměti, touto spotřebou si

vytvořil náskok o téměř 1,9GB před druhým nejméně náročným parserem, kterým byl Saxon se spotřebou lehce přes 3GB paměti.

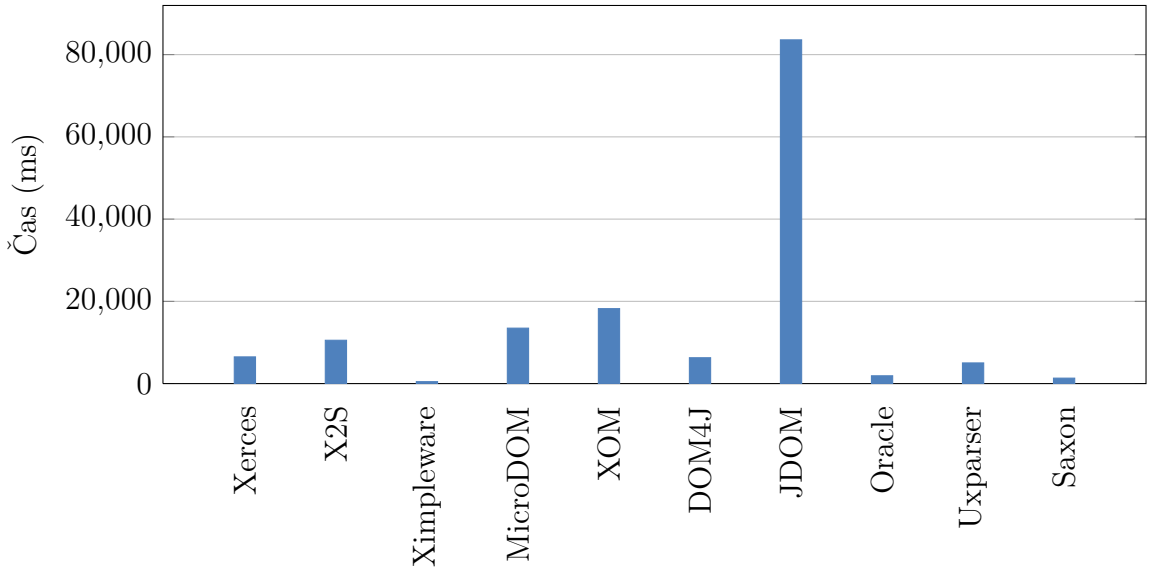
Ximpleware parser dosáhl opět nejvyšší přenosové rychlosti a sice 66 678kB/s. Druhým nejrychlejším parserem byl Oracle DOM s rychlostí 36 323kB/s. Jako zdaleka nejpomalejší skončil parser X2S s pouhými 128kB/s skončil daleko za 9. nejrychlejším parserem, kterým byl MicroDOM s rychlostí 13 195kB/s.

Celkové zhodnocení: Testování tohoto souboru dopadlo podobně jako u předchozích testů samostatných souborů, tedy Ximpleware parser dosáhl nejvyších rychlostí a měl nejmenší paměťové požadavky. X2S parser byl opět s přehledem nejpomalejší byl stotřikrát pomalejší než druhý nejpomalejší parser MicroDOM. MicroDOM byl opět paměťově nejnáročnějším parsersem s téměř dvojnásobnou paměťovou náročností oproti druhému nejnáročnějšímu parseru kterým byl JDOM.

6.6 Soubor čísel

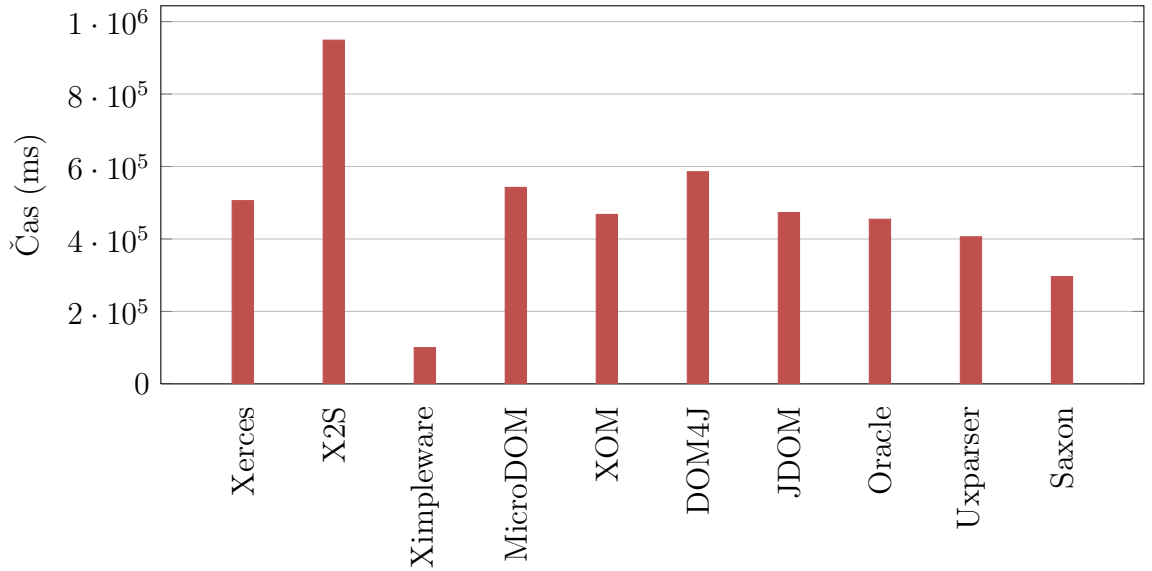
Jméno parseru	**Doba zpracování(ms)**	**Využitá paměť(kB)**	**BitRate (kB/s)**
Xerces	6 499	505 809	6 712
X2S	10 543	948 740	4 137
Ximpleware	474	100 097	92 027
MicroDOM	13 487	542 383	3 234
XOM	18 231	467 753	2 393
DOM4J	6 293	585 613	6 932
JDOM	83 591	472 960	522
Oracle DOM	1 902	454 500	22 934
Uxparser	5 031	405 981	8 670
Saxon	1 331	296 265	32 773

Tabulka 6.15: Výsledky měření: Soubor Čísel



Obrázek 6.25: Soubor Čísel: Doba zpracování

Tento soubor nejdéle zpracovával parser JDOM, který na zpracování souboru potřeboval 83 591ms, všechny ostatní parsery potřebovali na dokončení parsování méně než 20 vteřin. Druhým nejpomalejším parserem byl XOM s časem 18 231ms za ním následoval Micro-DOM s 13 487ms. Tradičně nejpomalejší parser X2S tentokrát potřeboval pouze 10 543ms na dokončení práce. Nejrychlejším parserem byl tradičně Ximpleware s 474ms následovaný Saxonem, který potřeboval 1 331ms.



Obrázek 6.26: Soubor Čísel: Využitá paměť

Nejméně paměti na zpracování souboru potřeboval Ximpleware(100 097), který potřeboval téměř o 200MB paměti méně než Saxon, který s 296 265kB skončil jako druhý, třetí

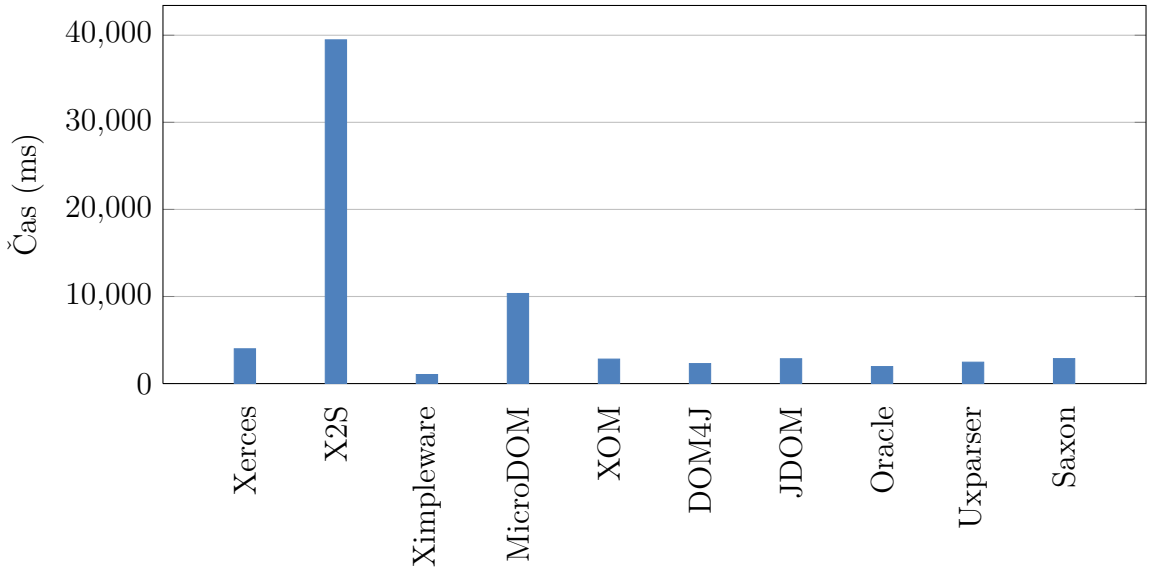
nejméně náročný Uxparser potřeboval 405 981kB paměti. Naopak nejvíce paměti spotřeboval parser X2S, který potřeboval 948 740kB paměti, za ním s propadem o 363 127kB následoval DOM4J. Kvůli enormně dlouhé době parsování se parser JDOM propadl na poslední místo v žebříčku propustnosti s pouhými 522kB/s. Druhý nejpomalejší byl XOM parser s rychlostí 2 393kB/s. Parserem s nejvyšší propustností byl opět Ximpleware, který dosáhl rychlosti 92MB/s.

Celkové zhodnocení: I v tomto testu Ximpleware potvrdil své prvenství. X2S parser skočil v žebříčku doby zpracování z 10. na 6. místo kde jej vystřídal JDOM, nicméně se X2S propadl v testu paměťové náročnosti a vystřídal na posledním místě MicroDOM parser, který byl tentokrát 8.

6.7 SwissProt

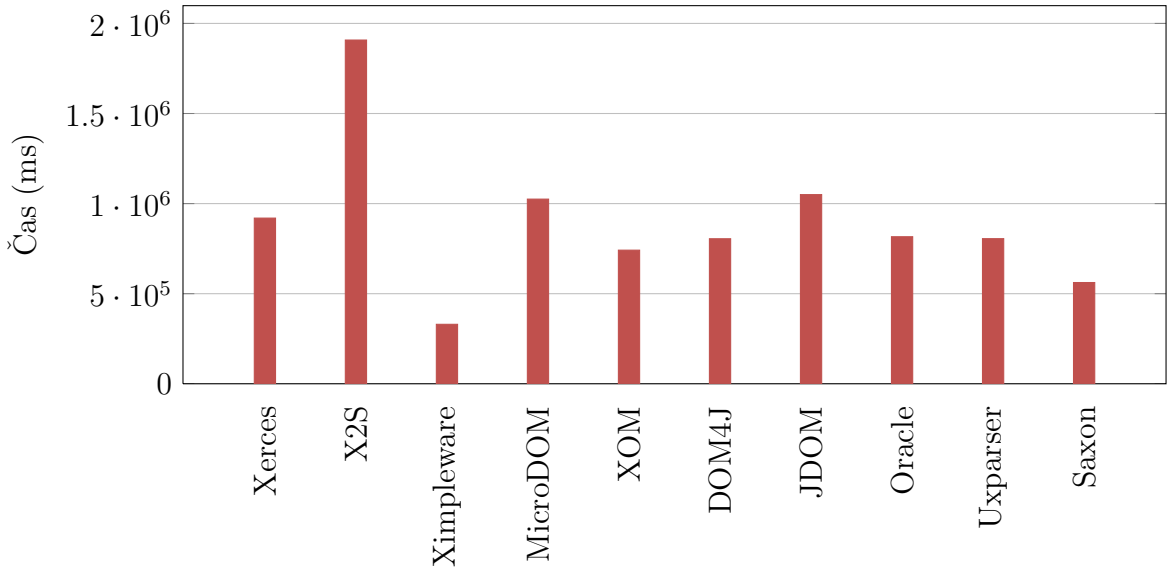
Jméno parseru	**Doba zpracování(ms)**	**Využitá paměť(kB)**	**BitRate (kB/s)**
Xerces	3 994	919 582	28 074
X2S	39 471	1 907 775	2 841
Ximpleware	1 040	330 167	107 817
MicroDOM	10 344	1 024 927	10 840
XOM	2 810	741 690	39 904
DOM4J	2 295	804 997	48 858
JDOM	2 851	1 049 931	39 330
Oracle DOM	1 953	816 197	57 414
Uxparser	2 456	805 424	45 655
Saxon	2 865	561 973	39 138

Tabulka 6.16: Výsledky měření: SwissProt



Obrázek 6.27: SwissProt: Doba Zpracování

Nejrychleji ze všech parserů opět parsoval Ximpleware, který dosáhl času 1 040ms za ním následoval s minimálním zpožděním Oracle DOM parser s časem 1 953ms. Většina zbývajících parserů se dostala pod 3 vteřiny, výjimkou je nejpomalejší trojice složená s parserů Xerces(3 994ms), MicroDOM(10 344ms) a X2S(39 471ms).



Obrázek 6.28: SwissProt: Využitá paměť

Ximpleware parser měl nejmenší paměťové požadavky, pro práci potřeboval 330 167kB, za ním následoval Saxon s 561 973kB. Krom tří paměťově nějnáročnějších parserů nepotřeboval žádný parser více než 1 000 000kB, tomuto číslo se ze spodu nejvíce blížil Xerces parser s požadavkem na 919 582kB paměti. Nejnáročnějším byl X2S parser, který

potřeboval 1 906 775kB paměti. Za parserem X2S následoval s druhou největší spotřebou paměti parser JDOM, který spotřeboval 1 049 931kB paměti.

Přenosová rychlost parseru Ximpleware jako jediná přesáhla rychlost 100MB/s. Jako druhý s propadem 50MB/s skončil parser Oracle DOM, který dosáhl rychlosti 57 414kB/s. Pod rychlost 10MB/s se dostal jako jedný parser X2S s rychlostí pouhých 2 841kB/s.

Celkové zhodnocení: X2S se opět prokázal jako výrazně nejpomalejší parser, druhý nejpomalejší parser MicroDOM měl před X2S parserem téměř trojnásobný náskok. Podobná situace nastala i u měření paměti, X2S potřeboval téměř dvojnásobek paměti než druhý nejnáročnější parser JDOM. Ximpleware byl už tradičně nejrychlejší a nejúspornější ze všech parserů s cca o 1,87násobkem přenosové rychlosti oproti Oracle DOM parseru který skončil na druhém místě.

Kapitola 7

Závěr

Tato práce měla za úkol otestovat a porovnat parsery využívající stromovou strukturu elementů. V této práci bylo testováno celkem 10 parserů, které byly testovány na stejném počtu kolekcí. Kolekce byly co možná nejvíce varibialní od jednoduchého číselníku, přes přepisy scénářů slavných divadelních her, údaje o amerických datových centrech a geoprostorové údaje až po národní a mezinárodní proteinové databáze či kolekce o desetitisích souborech.

U každého z testovaných parserů byly sledovány následující údaje:

1. Doba po kterou je soubor parsován
2. Kolik potřebuje k dané operaci paměti
3. Jak velký je výsledný tok dat

Ze všech testovaných parserů byl ve sledovaných kategoriích prokazatelně nejlepší parser Ximpleware, který byl ve všech testech nejrychlejší a měl nejvyšší propustnost a zároveň měl v 6/12 testů i nejnižší paměťové nároky. Nejhorší z testovaných parserů jsou bezesporu parsery X2S a MicroDOM, oba parsery se pravidělně řadili na nejnižších příčkách u testů propustnosti a naopak jejich nároky na paměť byly často mezi nejvyššími, stejně tak dosahované časy. Práce se měla původně zabývat pouze čistými DOM parsery, nicméně jak již bylo naznačeno v sekci 4.1 mnoho parserů kombinuje DOM a SAX přístup. Tato kombinace je zapřičiněna paměťovými požadavky všech DOM, kde potřebná kapacita paměti nezřídka kdy přesáhne až desetinásobek velikosti souboru. Z tohoto hlediska je nejvhodnější parsovat pomocí DOM parserů soubory o velikosti maximálně několik desítek MB.

Kapitola 8

Reference

[1] DOM, W3C
http://www.w3.org/DOM/

[2] W3C
http://www.w3.org/

[3] Wkipedia The Free Encyclopedia
https://en.wikipedia.org/wiki/Main_Page

[4] AbcLinuxu
http://www.abclinuxu.cz/

[5] RSS, Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/RSS

[6] RDF, W3C
http://www.w3.org/RDF/

[7] Parsery a XML aplikace, Téměř Vše o WWW
http://www.kosek.cz/clanky/swn-xml/ar02s30.html

[8] MathML,
http://en.wikipedia.org/wiki/Mathml

[9] SVG, Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/SVG

[10] XMPP, Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/XMPP

[11] SOAP, Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/SOAP

[12] SMIL, Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/SMIL

[13] WML, Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/Wireless_Markup_Language

[14] DocBook, Wikipedia The Free Encyclopedia

[15] Xpath, W3 schools
http://www.w3schools.com/xpath/

[16] Iliblio The Public's Library and Digital Archive
http://www.ibiblio.org/xml/examples/shakespeare/

[17] EPA United States Environmental Protection Agency
http://www.epa.gov/enviro/geo_data.html

[18] Data.gov Empowering people
https://explore.data.gov/Federal-Government-Finances-and-Employment/Federal-Data-Center-Consolidation-Initiative-FDCCI/d5wm-4c37/2

[19] Oxford Journals Briefings in Bioinformatics
http://bib.oxfordjournals.org/content/3/3/275.abstract

[20] PIR Protein Information Resource
http://pir.georgetown.edu/pirwww/dbinfo/pir_psd.shtml

[21] DTD, Téměř Vše o WWW
http://www.kosek.cz/clanky/swn-xml/dtd.html

[22] Root.cz
http://www.root.cz/

[23] **Kosek, Jiří**, Téměř Vše o WWW
http://www.kosek.cz/index.html

[24] W3 schools
http://www.w3schools.com/

[25] The Apache Software Foundation
apache.org

[26] StackOvweflor
stackoverflow.com

[27] SourceForge
http://sourceforge.net/

[28] HTML, W3 schools
http://www.w3schools.com/html/html_intro.asp

[29] XOM
http://www.xom.nu/

[30] JfreeChart
http://www.jfree.org/jfreechart/

[31] XHTML, W3 schools
http://www.w3schools.com/html/html_xhtml.asp

[32] Java-Source.net
http://java-source.net/open-source/xml-parsers

[33] File Guru
http://www.fileguru.com/Java-Micro-XML-Parser/info

[34] JDOM
www.jdom.org

[35] Balisage: The Markup Conferenece 2010
http://www.balisage.net/Proceedings/vol5/html/Probst01/BalisageVol5-Probst01.html

[36] **Jurek, Jan** Testování SAX Parserů, VŠB-TUO, 2011

[37] DOM4J
http://dom4j.sourceforge.net/dom4j-1.6.1/index.html

[38] MicroDOM
http://www.w3.org/TR/SVGTiny12/svgudom.html#Introduction

[39] Java API for XML Processing
https://jaxp.java.net/1.4/

[40] Saxonica
http://www.saxonica.com/welcome/welcome.xml

[41] Java Micro XML parser(Uxparser)
http://uxparser.sourceforge.net/

[42] Ximpleware
http://www.ximpleware.com/

[43] X2S
http://sourceforge.net/projects/x2s/

[44] Xerces
http://xerces.apache.org/

[45] CPAN
http://search.cpan.org/dist/XML-Bare/Bare.pm

[46] Chilkat software
http://www.chilkatsoft.com/java.asp

[47] CougarXML
http://www.cougarxml.com/

[48] Crimson
http://xml.apache.org/crimson/

[49] MSXML
http://msdn.microsoft.com/en-us/library/windows/desktop/ms763742%28v=vs.85%29.aspx

[50] TinyXML
http://www.grinninglizard.com/tinyxml/

[51] Inex-Wiki
http://db.cs.vsb.cz/files/inex-wiki.zip

[52] PIR-PSD
http://db.cs.vsb.cz/files/proteins.zip

[53] SwissProt
http://db.cs.vsb.cz/files/SwissProt.zip

[54] NanoXML
http://nanoxml.sourceforge.net/orig/

[55] W3C
http://www.w3.org/standards/xml/

[56] Isgmlug
http://www.isgmlug.org/

[57] GML, Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/IBM_Generalized_Markup_Language

Kapitola 9

Přílohy

Adresář	Obsah adresáře
app	Obsahuje testovací aplikace v jar
text	Bakalářská práce v pdf
lib	Obsahuje všechny použité XML knihovny
vys	Obsahuje všechny naměřené výsledky ve formátu csv